INSTITUT FÜR BIBLIOTHEKS- UND INFORMATIONSWISSENSCHAFT

Anforderungen, Entwurfsmuster und Systeme für kollaboratives Schreiben

D i s s e r t a t i o n

zur Erlangung des akademischen Grades

D o c t o r p h i l o s o p h i a e (Dr. phil.)

eingereicht an der

Philosophischen Fakultät I

der Humboldt – Universität zu Berlin

von

Frank Stüber

Dekan der Philosophischen Fakultät I:

Prof. Dr. Michael Seadle

Gutachter:	1. Prof. Dr. Peter Schirmbacher
	2. Prof. Dr. Wolfgang Coy
Datum der Disputation:	13. Oktober 2011

Vorwort

Ich möchte mich an dieser Stelle bei all denjenigen bedanken, die ihren Beitrag zum Entstehen dieser Arbeit geleistet haben.

An erster Stelle möchte ich Herrn Prof. Peter Schirmbacher meinen Dank aussprechen, der es mir ermöglicht hat, diese Dissertation am Institut für Bibliotheks- und Informationswissenschaft der Humboldt-Universität Berlin zu schreiben. Er war während der gesamten Zeit stets ein äußerst angenehmer Rat- und Ideengeber.

Des Weiteren gilt mein Dank Herrn Prof. Wolfgang Coy, Frau Juliane Schiel, Frau Christa Nowakowski sowie allen Teilnehmern des Doktorandenkollegs.

Mein besonderer Dank gilt meiner Familie, insbesondere meinen Eltern, welche im weiteren Sinne die entscheidenden Voraussetzungen für diese Arbeit gelegt haben, sowie meinen beiden Brüdern und meiner Frau Lidiya, die mir alle jene Unterstützung haben zukommen lassen, ohne die ein Abenteuer dieser Art nicht möglich gewesen wäre.

Inhaltsverzeichnis

1 Einführung

Kollaboratives Schreiben bezeichnet das gemeinsame Erstellen eines Textdokuments in einer Gruppe, deren Mitglieder zur gleichen Zeit oder an unterschiedlichen Zeiten, am gleichen oder an unterschiedlichen Orten arbeiten können.

Im Rahmen der Erforschung der Auswirkungen des Klimawandels auf die Umwelt hat das von der *World Meteorological Organization* (WMO) und dem *United Nations Environment Programme* (UNEP) ins Leben gerufene *Intergovernmental Panel on Climate Change* (IPCC) in den letzten Jahren eine Reihe von Berichten veröffentlicht, darunter den IPCC WG1 AR4 Report aus dem Jahre 2007. Insgesamt haben über 650 Wissenschaftler aus allen Teilen der Welt zum Inhalt des Dokumentes beigetragen, hinzu kommen noch über 200 Gutachter. Dieses Beispiel zeigt, was kollaboratives Schreiben im Extremfall bedeuten kann. Ein Großteil aller erstellten Textdokumente weisen heutzutage mehr als einen Autor aus [Ede et al. 90]. Schon die Entstehung der Bibel war das Ergebnis einer Multiautorenschaft. Selbst in der Belletristik ist eine Multiautorenschaft keine Seltenheit. Als Beispiel sei hier der Kriminalroman „Machtspiele“ der drei französischen Autoren Daniel Pennac, Jean-Bernard Pouy und Patrick Raynal aus dem Jahre 1985 erwähnt, bei dem alle drei Autoren auf recht ungewöhnliche Weise kollaborieren, in dem sie ihren jeweils individuellen Schreibstil bewusst beibehalten und nicht homogenisiert haben.

Das kollaborative Schreiben ist ein Teilaspekt des kollaborativen Arbeitens (Collaborative Work) zu dessen breit gefächertem Instrumentarium eine Vielzahl von Software-Systemen gehören, die auch oft unter dem Stichwort Groupware zusammengefasst werden. Beispiele hierfür sind:

1. ***E-Mail*** als einen der ältesten und beständigsten Software-Dienste zum gemeinsamen Austausch von Nachrichten, Dokumenten und Informationen.

2. ***Chat und Video-Conferencing*** zur Echtzeitkommunikation, häufig miteinander verknüpft (siehe z.B. Skype).

3. ***Zentrale Informationssysteme***, die einer Gruppe von Nutzern eine beliebige Menge von gemeinsam nutzbaren Ressourcen (Kontakte, Termine, Dokumente etc.) zu Verfügung stellen (z.B. Microsoft Exchange).

4. ***Soziale Plattformen***, die einer Gruppe von Teilnehmern unterschiedliche Möglichkeiten bieten, miteinander in Kontakt zu treten, um Gefühle, Ideen, Meinungen, Informationen untereinander auszutauschen.

Kollaboratives Schreiben umfasst organisatorische, soziale und vor allem technologische Aspekte. Sind Autoren auf unterschiedliche Orte und vielleicht noch Zeitzonen verteilt, benötigen sie Möglichkeiten, um sich untereinander zu synchronisieren. Dazu gehören Kommunikationswerkzeuge wie E-Mail, Chat, Video-Konferenz und das gute alte Telefon. Ein wesentlich effizienteres Werkzeug stellt jedoch eine

gemeinsame Software zum Schreiben, Verwalten, Austauschen und Beobachten eines Textes dar. Entsprechende Software-Implementationen sind jedoch rar gesät. Die klassischen Textverarbeitungssysteme sind auf die Unterstützung von Einzelautoren ausgelegt. Möchten mehrere Autoren am selben Text arbeiten, müssen sie sich untereinander abstimmten in Hinblick auf Inhalt, Layout und Zeit. Je größer die Anzahl der Autoren desto größer der Abstimmungsaufwand. Kommen weitere Akteure ins Spiel, die zwar nicht direkt schreiben, aber dennoch Zugriff auf den Text benötigen, wie zum Beispiel Teamleiter, Gutachter oder Editoren, nimmt der Kollaborationsbedarf noch weiter zu.

Dies steht im Kontrast zur aktuellen Entwicklung im Word Wide Web. Unter dem Stichwort Web 2.0 [O'Reilly 05] haben sich in den letzten Jahren eine Reihe von Diensten etabliert, deren gemeinsames Merkmal es ist, dass sie eine direkte Kollaboration zwischen mehreren Teilnehmern über das Internet ermöglichen. Eines der prominentesten Beispiele stellt das Internetangebot von Wikipedia da, einer frei zugänglichen Online-Enzyklopädie in verschiedenen Sprachen. Laut Eigendarstellung zählt Wikipedia mittlerweile zu den zehn am häufigsten frequentierten Internetangeboten der Welt. Allein die englischsprachige Ausgabe von Wikipedia stellt über zwei Millionen Artikel zur Verfügung. Wikipedia erlaubt es jedem Besucher, die Inhalte von Artikeln direkt in einem Internet-Browser zu redigieren oder neue Artikel zu verfassen. Eine ähnliche rasante Entwicklung im Bereich der klassischen Textverarbeitung ist nicht erkennbar, obwohl es in den letzten Jahren zahlreiche Neuerungen auf dem Gebiet der Online-Textverarbeitung gegeben hat, sowie Forschungsprojekte, um die kollaborativen Fähigkeiten von Textverarbeitungssystemen zu verbessern. Ein interdisziplinäres Projekt der Berliner Humboldt-Universität (siehe Kapitel 2.2.1), bei dem insgesamt 26 Historiker in direkter Zusammenarbeit ein gemeinsames Buch geschrieben haben, zeigt die Problematik in der Praxis auf. Die Suche nach einem geeigneten Software-Werkzeug im Vorfeld dieses Projektes kam zum Ergebnis, dass lediglich die von Wikipedia eingesetzte Basistechnologie (ein Wiki-System) das nötige Potential zur technischen Umsetzung besaß, obwohl sie eigentlich nicht für das kollaborative Schreiben von Büchern konzipiert ist.

Diese Dissertation möchte daher einen Beitrag dazu leisten, den Paradigmenwechsel in der Textverarbeitung zu beschleunigen, weg vom Einzelautor und hin zur Multiautorenschaft. Die wissenschaftliche Fragestellung lässt sich zusammengefasst wie folgt formulieren:

1. Welches Anforderungsprofil ergibt sich für eine kollaborative Textverarbeitung?

2. Welche alternativen oder auch sich ergänzende technische Möglichkeiten (Entwurfsmuster) stehen für die Unterstützung eines solchen Profils zur Verfügung?

3. Was sind die sich aus beiden oben genannten Punkten ergebenden Realisierungsoptionen für eine idealtypische kollaborative Textverarbeitung und wie können diese umgesetzt werden?

Die Arbeit konzentriert sich dabei auf das kollaborative Erstellen wissenschaftlicher bzw. technischer Texte, d.h. auf einen Bereich, in dem kollaboratives Schreiben häufiger vorkommt als etwa in der Belletristik. Die Diskussionen und Ergebnisse dieser Arbeit lassen sich aber analog auf beliebige andere Textgattungen übertragen.

In Kapitel 2 „Kollaboratives Schreiben (KS)“ wird unabhängig von einer bestimmten Software in die Thematik des kollaborativen Schreibens eingeführt. Neben Beispielen aus Wissenschaft und Wirtschaft wird in diesem Kapitel eine allgemeingültige Taxonomie für kollaboratives Schreiben vorgestellt.

In Kapitel 3 „KS-Software“ werden Textverarbeitungssysteme auf ihre kollaborativen Fähigkeiten hin untersucht. Im Anschluss daran werden Anforderungen an ein kollaboratives Textverarbeitungssystem formuliert und zwei populäre Vertreter für kollaboratives Schreiben, Google Text und Microsoft Word, analysiert.

In Kapitel 4 „KS-Entwurfsmuster“ werden allgemeingültige Entwurfsmuster von Software-Werkzeugen zur Unterstützung des kollaborativen Schreibens herausgearbeitet, auf die dann im Kapitel 5 Bezug genommen wird.

In Kapitel 5 „KS-Software-Architekturen“ werden schließlich zwei spezielle Lösungsansätze für wissenschaftliche KS-Systeme beschrieben, welche aufgrund der vorhergehenden Überlegungen konkrete Vorschläge für eine spätere Implementation machen. Der der erste Ansatz beschreibt eine implementationsunabhängige Erweiterung eines etablierten Netzwerkprotokolls um zusätzliche kollaborative Fähigkeiten. Der zweite Ansatz ist eine Architekturvorgabe für das Organisieren eines kollaborativen Schreibprojektes innerhalb eines Softwaresystems, dessen konkrete Implementation Teil weiterführender Arbeiten sein könnte.

2 Kollaboratives Schreiben (KS)

2.1 Einführung

Während sich die Frage „Was ist kollaboratives Schreiben?“ im ersten Moment recht intuitiv beantworten lässt („Gemeinsame Erstellen eines Textes in einer Gruppe“), stellen sich bei näherer Betrachtung der Thematik doch sehr viel mehr Fragen. Geht es wirklich nur um das Schreiben oder ist nicht auch das Planen des Inhalts, das Korrekturlesen, die Diskussion über einzelne Textpassagen oder die anschließende Publikation ein Teil dieses Prozesses? Wo beginnt und wo endet kollaboratives Schreiben? Ist die angebotene Hilfe beim Verfassen eines Textes schon und das Schreiben eines Zeitungsartikels noch kollaboratives Schreiben? Wie schnell kollaboratives Schreiben an Komplexität gewinnen kann, soll das folgende kleine Beispiel aufzeigen: Zwei Autoren A und B möchten gemeinsam einen Artikel schreiben. Die Frage, die sich beiden stellt, ist: Wie sollen sie sich organisieren? Es ergeben sich spontan die folgenden vier Möglichkeiten:

(a) Autor A diktiert und Autor B schreibt den Text.

(b) Autor A schreibt einen ersten Entwurf, der anschließend von Autor B gegengelesen und ggf. überarbeitet wird.

(c) Autor A schreibt erst den ersten Teil des Artikels und anschließend Autor B den zweiten Teil des Artikels

(d) Autor A schreibt den ersten Teil des Artikels während gleichzeitig Autor B den zweiten Teil schreibt.

Eine kurze Analyse der vier Optionen ergibt folgendes Ergebnis:

(a) ist die einfachste aller Möglichkeiten. Autor A kann den Text auf ein Diktiergerät sprechen, das er Autor B zum Abtippen aushändigt. Er kann sich auch mit Autor B an einen Tisch setzen, um ihm direkt den Text zu diktieren.

(b) ist schon etwas komplizierter. Möchte Autor B den Text von Autor A überarbeiten, so muss er die Möglichkeit haben, Korrekturen vornehmen zu können, Textpassagen zu löschen oder neue einzufügen. Gegebenenfalls müssen diese Änderungen vorher noch gemeinsam diskutiert werden.

(c) ist wieder etwas einfacher. Autor A muss lediglich den noch unfertigen Artikel an Autor B aushändigen, damit dieser ihn weiterschreiben kann.

(d) ist am kompliziertesten. Zunächst müssen sich Autor A und Autor B darauf einigen, welchen Teil des Textes der eine und welchen Teil der andere zu schreiben hat und wo exakt die Schnittstelle zwischen beiden Teilen liegt. Anschließend

müssen sie sich überlegen, wie die jeweiligen Teiltexte am Ende zu einem ganzen Text zusammengefügt werden können.

Aus diesem Beispiel lassen sich neben der Tatsache, dass es stets mehrere Möglichkeiten gibt, einen Text gemeinsam zu verfassen, auch noch folgende Beobachtungen ableiten: In allen vier Handlungsalternativen müssen Autor A und Autor B miteinander kommunizieren. In (a) muss Autor A diktieren und Autor B zuhören, in (b) und (c) muss Autor A den Artikel an Autor B übergeben, in (d) müssen beide Autoren gemeinsam planen. In (b) – (d) müssen beide Autoren ihre Textbeiträge in irgendeiner Form synchronisieren, so dass am Ende ein Gesamttext entstehen kann. Der zeitliche Aufwand scheint in (d) am geringsten zu sein, da beide Autoren parallel arbeiten und jeweils nur die Hälfte des Artikels zu schreiben haben.

Wenn man jetzt noch hinzunimmt, dass Texte sehr viel umfangreicher und komplexer als ein Artikel seien können und dass mehr als zwei Autoren an der Entstehung eines Textes beteiligt sein können, dann ist man schon mitten im Forschungsgebiet *Kollaboratives Schreiben*. Arbeiten zu diesem Thema reichen zurück bis in die späten 80er und frühen 90er Jahre. Die überwiegende Anzahl der Arbeiten versuchen an Hand von Feldversuchen sowie durch Interviews ein besseres Verständnis für die Thematik zu erhalten, um anschließend Anforderungen für Software-Werkzeuge abzuleiten. Andere Arbeiten versuchen vorhandene Software-Werkzeuge zu evaluieren und auf ihre Tauglichkeit für kollaboratives Schreiben zu untersuchen.

Zunächst jedoch soll das kollaborative Schreiben unabhängig von einer bestimmten Software bzw. Software-Gruppe betrachtet werden, auch wenn eine völlige Ausblendung von Software kaum möglich ist – immerhin wird heute nur noch in sehr seltenen Fällen auf die Schreibmaschine oder auf Papier und Bleistift zurückgegriffen. In den folgenden beiden Kapiteln wird Textkollaborationen sowohl im wissenschaftlichen wie auch im unternehmerischen Umfeld betrachtet und analysiert. Das Ziel hierbei ist es, die Praxisrelevanz von Textkollaboration herauszuarbeiten.

Das nächste Kapitel baut auf einen wissenschaftlichen Artikel von [Lowry et al. 04] auf und versucht die verschiedenen Aspekte Textkollaboration zu formalisieren. Das Ziel hierbei ist es, eine möglichst allgemeingültige Begrifflichkeit für die weitere Verwendung innerhalb dieser Arbeit zu definieren.

2.2 Textkollaboration in der Wissenschaft

2.2.1 Beispiel

Die Deutsche Forschungsgemeinschaft hat im Mai 2004 die Einrichtung des Schwerpunktprogramms 1173 „Integration und Desintegration der Kulturen im europäischen Mittelalter" für sechs Jahre (2005-2011) beschlossen. Eines der Ergebnisse der ersten Förderphase im Jahr 2007 war ein Buch mit dem Titel „Mittelalter im Labor: Die Mediävistik testet Wege zu einer transkulturellen Europawissenschaft", das gleichzeitig in klassischer Papierform und als Hypertext im Internet [Schiel et al. 08] veröffentlicht wurde. Das interessante an dieser Veröffentlichung ist nun, dass

der Text durch eine kollaborative Zusammenarbeit von 26 Historikern aus ganz Deutschland entstanden ist. Die technische Grundlage für das praktizierte kollaborative Schreiben war eine abgepasste Version von MediaWiki, jenem Wiki-System, mit dem auch die Online-Enzyklopädie Wikipedia erstellt wird. Gespräche mit zwei Beteiligten an diesem Projekt haben einige interessante Aspekte zu Tage gefördert. Positive Aspekte waren:

- Durch die Einbindung aller in ein Team hat sich eine sehr enge Form der Zusammenarbeit gebildet, da sich die Autoren untereinander absprechen mussten in Hinblick auf Organisation, Ziele, Zeiten und Werkzeuge.

- Dies hat gleichzeitig durch Abgleich mit inhaltlichen, theoretischen und methodologischen Ansätzen der anderen beteiligten Disziplinen zu einer Förderung von neuen Ideen geführt.

- Da diese Form der Kooperation, dazu noch im geisteswissenschaftlichen Bereich, sehr ungewöhnlich war, hat das gesamte Projekt automatisch eine höhere Aufmerksamkeit in der Öffentlichkeit generiert.

Es wurden aber auch einige psychologische Hemmnisse genannt. Für viele Autoren war diese Form der Zusammenarbeit eine mehr oder weniger große Umstellung. Der Einzelne ist nun Teil einer Autorengemeinschaft. Texte können von Kollegen korrigiert, geändert oder verworfen werden. Und damit ist die Urheberschaft eines Textes bzw. eines Textabschnittes nicht mehr in jedem Fall einem Einzelnen zuordenbar. Als potentielle Problemstellen bei der Arbeit mit einem kollaborativen Software-Werkzeug wurden die folgenden Punkte genannt:

- Texte besitzen unterschiedliche inhaltliche Anforderungen wie das Einbinden von Formeln, Grafiken, Bildern und Videos.

- Texte besitzen unterschiedliche formale Anforderungen wie das Erstellen von Verzeichnissen (z.B. Inhaltsverzeichnisse, Abbildungsverzeichnisse, Indexe, etc.) oder die spezielle Auszeichnung von Teiltexten (z.B. Zitate, etc.).

- Texte können mehrere Publikationsformen aufweisen (z.B. Buch und Web).

- Es ist üblich und auch gewollt, dass Teammitglieder nicht immer am gleichen Ort ihre Texte verfassen. Es gibt beispielsweise Zeiten, an denen im Büro geschrieben wird und es gibt Zeiten, zu denen zu Hause oder unterwegs geschrieben wird.

- Die technische Ausstattung von Teammitgliedern kann sehr stark variieren. Viele hatten zu Hause keinen oder nur einen sehr langsamen Internetanschluss.

- Alle Beteiligten hatten zum Teil sehr unterschiedliche Erfahrungen und Gewohnheiten was Textverarbeitungssysteme angeht.

- In Arbeitsphasen mit hohem Zeitdruck (z.B. bedingt durch Redaktionsschlüsse) ist die Zuverlässigkeit des genutzten Systems von großer Bedeutung.

Insgesamt wurde die Arbeit mit dem Wiki-System als positiv bewertet, unter anderem aber auch auf Grund mangelnder Software-Alternativen.

2.2.2 Wissenschaftliches Publizieren

In Verallgemeinerung eines Mottos aus [Ebel et al. 06] sei angemerkt:

> *Was immer wissenschaftlich entwickelt, theoretisiert, gefunden, erfunden, gemessen wird – es verdient nicht, entdeckt zu werden, wenn es anderen nicht mitgeteilt wird.*

Dazu haben sich seit den ersten Papyrusrollen eine Reihe von Formen entwickelt, geschriebenes Wissen anderen mitzuteilen. Eine wissenschaftliche Publikation ist in der Regel der Endpunkt einer vorangegangenen Forschungstätigkeit, deren Ergebnisse nun durch ihre Publikation der Öffentlichkeit zugänglich gemacht werden. Publikationen dieser Art besitzen einen hohen Wert für die beteiligten Autoren, da sie ihr Renommee fördern bzw. zu einer unmittelbaren Auszeichnung (z.B. Diplom) führen können. Dementsprechend verfolgt jeder Autor auch immer ein gewisses Eigeninteresse, das nicht immer in Einklang mit dem Team als Ganzes zu bekommen ist. Hinzu kommt, dass wissenschaftliche Texte bestimmten Formanforderungen genügen müssen und im Extremfall einen gedruckten Umfang von mehreren hundert Seiten erreichen können. All dies hat zur Folge, dass wissenschaftliche Textkollaboration nur durch eine feine Ausbalancierung der zum Teil konkurrierenden Interessen aller Akteure möglich ist.

Unabhängig davon, ob die Publikation elektronisch oder per Papier erfolgt, orientieren sich die Publikationsformen derzeit in der Regel noch an der Publikation per Papier, die sich in folgende Kategorien unterscheiden lässt:

Berichte – Ein Bericht ist ein wissenschaftlicher Text, der die Ergebnisse eines Forschungsprozesses strukturiert und konsistent zusammenfasst. Der Bericht ist ein dauerhaftes in sich abgeschlossenes Dokument. Ein Bericht muss in der Regel bestimmten Formen genügen (z.B. darf ein wissenschaftlicher Artikel für ein wissenschaftliches Journal eine bestimmte Anzahl von Worten nicht überschreiten), eine Zusammenfassung, ein Inhalts- und Stichwortverzeichnis und ein Literaturverzeichnis haben. Ein Bericht entsteht in einem wissenschaftlichen Zusammenhang, der durch den Inhalt und Literaturverweise erkennbar sein soll. Ein Bericht enthält eine eindeutige Identifizierung durch Titel, Angabe des Datums, an dem der Bericht abgeschlossen bzw. erstellt wurde und die Angabe des Autors bzw. der Autoren. Es lassen sich verschiedene Arten von Berichten je nach Umfeld unterscheiden, z.B. Laborberichte, Diplom-, Bachelor-, Seminar-, Praktikums-, Abschluss-, Doktorarbeiten, Forschungsanträge, Gutachten.

Labor- und Protokollbücher – Ein Laborbuch ist ein Tagebuch des experimentierenden Naturwissenschaftlers, in dem der Fortschritt und das Ergebnis wissenschaftli-

cher Experimente in einer Laborumgebung von einem oder mehreren Wissenschaftlern festgehalten werden. Das Gegenstück zum Laborbuch außerhalb eines Laborumfeldes ist das Protokollbuch, in dem z.B. Biologen, Archäologen, Geologen ihre Befunde festhalten.

Fachzeitschriften – In Fachzeitschriften werden wissenschaftliche Berichte (Artikel) zusammengefasst, in denen die Ergebnisse wissenschaftlicher Arbeit beschrieben werden, um sie dem Fachpublikum mitzuteilen. Ein wissenschaftliches Ergebnis gilt dann als veröffentlicht (publiziert).

Fachbücher – Ein Fachbuch kann als ein sehr umfangreicher Bericht betrachtet werden, d.h. es umfasst in der Regel mehr Seiten als ein Bericht und erlaubt einen wissenschaftlichen Sachverhalt, eine Theorie bzw. einen Prozess umfassender darzustellen. Der wesentliche Unterschied zu einem Bericht besteht jedoch darin, dass das Buch von einem Verlage verlegt worden ist.

Elektronische Publikationsformen – Elektronische Publikationen können zum einen darin bestehen, ein elektronisches Gegenstück des Papierdokuments zu veröffentlichen, z.B. indem man einen Bericht als PDF-Dokument veröffentlicht. Andere elektronische Publikationsformen liefern z.B. Wiki-Systeme, Content-Management-Systeme (CMS), Blogs, die aber mit den gleichen traditionellen Metaphern arbeiten: Ein Wiki-System entspricht einem Lexikon, ein CMS arbeitet mit Artikeln, ein Blog basiert ebenfalls auf Artikeln oder kurzen Mitteilungen. Der charakteristischen Unterschiede elektronischer Publikationen gegenüber den Papierformen liegen in

- der Verfügbarkeit
- der Archivierbarkeit
- der Recherchierbarkeit
- der benutzerfreundlichen Navigation durch komplexe Sachverhalte aufgrund direkter Querverweise (Links) und eingebetteter Medien

Benötigt man bei Papierformen umfangreiche Archive und Bibliotheken, in denen man oft persönlich und vor Ort recherchieren muss, können elektronische Systeme ihre Inhalte per Internet jedem bzw. dem Fachpublikum überall dort verfügbar machen, wo ein Internetanschluss besteht (Open Access). Vor allem aber kann über Schlagworte, Taxonomien, Volltextsuche wesentlich detaillierter und breitgefächerter recherchiert werden. Dieser Vorteil ist gleichzeitig mit der Gefahr verbunden, dass die schiere Menge an Informationen, die auch durch die schnelleren Publikationszeiten elektronischer Medien begünstigt wird, eine weniger fokussierte und somit komplexere Darstellung von Forschungsergebnissen zur Folge hat. Direkte Querverweise (Links) und die Einbettung von Videos, Tondokumenten (z.B. der Originalaufnahme einer Rede) ermöglichen es, wissenschaftliche Texte um weitere Originalinformationen anzureichern, die gegenüber den Textzitaten der Papierformen eine neue Qualität darstellen. Zudem unterscheidet sich das Layout elektronischer Publikationen von Papierformen dadurch, dass die Seitengröße bedingt durch

das Bildschirmformat nicht mehr den Papierformaten entspricht und durch die direkten Querverweise (Links) die Inhalte nicht notwendigerweise mehr sequenziell bzw. mit hierarchischen Überschriften wie in einem Bericht, Artikel bzw. Buch (Titel, Zusammenfassung, Einleitung, Hauptteil, Schlussfolgerungen, Quellenverzeichnis) orientiert werden müssen.

Jede Publikationsform enthält Objekte aus einer vorgegebenen Menge von Klassen. Jede Objektklasse besitzt eigene Formanforderungen. In wissenschaftlichen Texten lassen sich in der Regel folgende Objektklassen finden:

- Gliederungen (Kapitel, Absätze, Aufzählungen)
- Verzeichnisse (Inhalt, Index, Abbildungen, Tabellen, Abkürzungen, Symbole, Literatur)
- Eingebettete Medien (Fotos, Grafiken, Videos, Ton)
- Zitate (Verweise auf Originalquellen)
- Eingebettete Originalquellen (Quelltexte, Auszüge aus Laborberichten)
- Aufzählungen
- Querverweise
- Tabellen
- Formeln
- Fußnoten

Einige Objektklassen sind selbstverständlich häufiger zu finden (z.B. Verzeichnisse) als andere (z.B. Formeln). Auch ist diese Auflistung mit Sicherheit nicht erschöpfend. Allerdings ist unmittelbar klar, dass eine wissenschaftliche Publikation Formanforderungen verlangt, deren Besonderheiten wiederum spezielle Anforderungen an das kollaborative Schreiben generieren. Als Beispiel sei das Zitieren genannt. Es gilt die Regel, dass ein Zitat nur im Original abgedruckt werden darf. Dieser Textteil darf daher auch bei dringendem Verdacht auf Fehler unter keinen Umständen von einem anderen Autor ohne Zustimmung des Originalautors geändert werden. Mit anderen Worten: Zitate müssen einer besonderen Form der Zugriffsberechtigung unterliegen. Ein Forderung, die bei Einzelautoren bedeutungslos wäre.

2.2.3 Wissenschaftliche Kollaboration

Bis in das letzte Jahrhundert hinein wurde Forschung vor allem durch weitestgehend isoliert arbeitende Wissenschaftler oder wissenschaftliche Einrichtungen betrieben. Dies hat sich in der Gegenwart grundlegend geändert. Die Gründe hierfür sind u.a.:

- Universitäten und Forschungseinrichtungen haben oftmals nicht das Geld, die Ausrüstung oder die Expertise, um die eigene Forschung auf höchstem Niveau vorantreiben zu können.

- Komplexe Forschungsfragen der heutigen Welt erfordern die – ggf. interdisziplinäre – Zusammenarbeit verschiedener Forschungsinstitutionen und Wissenschaftlergruppen. Als Beispiele seien hier die umfangreichen Projekte zum Klimawandel, zur Genforschung, zur Aidsforschung, zur Kernenergienutzung und zur physikalischen Grundlagenforschung genannt.

- Der Austausch von wissenschaftlichen Daten, Ergebnissen und Informationen über das Internet lässt die möglicherweise große geographische Entfernung von wissenschaftlichen Einrichtungen auf ein Minimum reduzieren. Das gleiche gilt in abgeschwächter Form auch für die Tatsache, dass ein Wissenschaftler heute mit relativ wenig Geld zu nahezu jeden Punkt der Erde fliegen kann.

- Junge Wissenschaftler sind mit sozialen Diensten wie Facebook oder YouTube aufgewachsen und stehen dadurch Kollaborationen im Allgemeinen sehr viel offener gegenüber als dies manchmal bei Wissenschaftlern der älteren Generation der Fall ist.

Wissenschaftliche Kollaboration hat unterschiedliche Ausprägungen. Es kann sich dabei um die lose Zusammenarbeit zweier Wissenschaftler handeln oder aber auch um ein multinationales Forschungsprojekt, wie sie zum Beispiel am *European Organization for Nuclear Research* (CERN) in der Schweiz durchgeführt werden, einer Einrichtung, bei der sich 20 Mitgliedstaaten finanziell und personell beteiligen. [Shrum et al. 2007] unterscheidet vier Typen von wissenschaftlicher Kollaboration:

- ***Bürokratische Kollaboration*** – Bürokratische Kollaboration ist wie der Name schon nahelegt durch hierarchische Autoritäten, geschriebene Regeln, formalisierte Verantwortung und spezialisierte Arbeitsteilung gekennzeichnet. Diese Form der Kollaboration kennzeichnet in der Regel komplexe wissenschaftliche Projekte.

- ***Führerlose Kollaboration*** – Führerlose Kollaboration ist wie bürokratische Kollaboration formalisiert, allerdings fehlt eine hierarchische Autorität, z.B. ein ausgezeichneter Wissenschaftler, der die Rolle des Teamleiters übernimmt.

- ***Nicht-spezialisierte Kollaboration*** – Nicht-spezialisierte Kollaboration ist das Komplement zur führerlosen Kollaboration. Es gibt zwar eine hierarchische Autorität, eine spezialisierte Arbeitsteilung wie in der bürokratischen Kollaboration existiert aber nicht.

- ***Partizipatorische Kollaboration*** – Partizipatorischer Kollaboration fehlt jede Arbeitsteilung und alle weiteren Merkmale der bürokratischen Kollaboration. Es arbeite also eine Gruppe von Spezialisten ohne nennenswerte Formalis-

men und ohne hierarchische Autoritäten zusammen. Als Beispiel dient Teilchenphysik-Projekt gegen Ende des Kalten Krieges: die beteiligten US-amerikanischen und sowjetischen Forscher konnten trotz der erheblichen Unterschiede in Ideologie und kulturellem Hintergrund wissenschaftliche Ideen und Kritik ohne nennenswerte Hindernisse austauschen.

In einem Punkt ähneln sich jedoch alle wissenschaftliche Kollaborationen: Die Ergebnisse müssen angemessen publiziert werden, ein Vorgang, der in der Regel auf eine Textkollaboration hinausläuft. Und genau dafür werden passende Software-Werkzeuge benötigt. Die von [Shrum et al. 2007] durchgeführte Typisierung weist dabei bereits auf organisatorische Rahmenbedingungen für kollaboratives Schreiben hin, die in Kapitel 2.4 weiter formalisiert werden.

2.3 Textkollaboration in Unternehmen

2.3.1 Beispiel

Im Jahre 2007 war ich selbst an einem Projekt beteiligt, bei dem zwei Firmen gemeinsam an einer europaweiten Ausschreibung für die Entwicklung einer Software für öffentliche Behörden teilgenommen haben. In diesem Zusammenhang wurde ein mehrere hundert Seiten großes Angebot, bestehend aus rechtlichen, technischen wie auch finanziellen Ausführungen, erstellt. Da kein Einzelautor die nötigen Kompetenzen für alle Bereiche des Angebotstextes hatte, musste die Arbeit auf beide Firmen bzw. auf mehrere Autoren aufgeteilt werden. Da zudem die Zeit für die Erstellung des Angebots begrenzt war, musste parallel am Text gearbeitet werden. Der praktizierte Workflow gestaltete sich wie folgt:

- Es wurde zunächst eine sogenannte Clearing-Stelle, bestehend aus zwei Autoren, die keine wesentlichen inhaltlichen Beiträge verfassen sollten, eingerichtet.

- Diese Clearing-Stelle bereitete in Absprache mit allen anderen Autoren ein minimales Grundgerüst für das resultierende Gesamtdokument vor. Dieses minimale Grundgerüst bestand aus einem vorläufigen Inhaltsverzeichnis, Platzhaltern für die einzelnen Kapitel sowie einigen wenigen bereits gefüllten Kapiteln (z.B. rechtliche Textpassagen, die von älteren Angebotstexten einfach übernommen wurden).

- Anschließend wurde untereinander abgesprochen, welche Autoren welche Kapitel zu schreiben haben. Dies wurde direkt im Text hinter den Kapitelüberschriften temporär vermerkt.

- Das aktuelle Gesamtdokument wurde nun per E-Mail an alle Autoren verschickt. Als Textverarbeitung wurde damals Microsoft Word 2003 eingesetzt.

- Jeder Autor schrieb nun innerhalb des Gesamtdokuments die Kapitel, für die er verantwortlich war und schickte das geänderte Gesamtdokument zu ei-

nem vorher abgesprochenen Zeitpunkt per E-Mail zurück an die Clearing-Stelle.

- Die Clearing-Stelle synchronisierte alle geänderten Texte in einem neuen Gesamtdokuments, das wiederum an alle Autoren per E-Mail versendet wurde, damit sie weiter an ihren Kapiteln arbeiten konnten.

- In der Zeit zwischen dem Einsammeln der geänderten Texte und dem Verteilen des synchronisierten Gesamtdokumentes durften keine weiteren Texte geschrieben werden.

- Dieser Zyklus von Schreiben, Einsammeln, manuellem Synchronisieren und erneutem Verteilen des Textes wurde so lange wiederholt, bis der Text inhaltlich seine endgültige Fassung erreicht hatte.

- Der Zyklus wurde durch regelmäßige persönliche Treffen unterbrochen, bei denen möglichst alle Autoren anwesend waren, um den bereits geschrieben Teil des Angebots gemeinsam am Bildschirm durchzugehen. Bei dieser Gelegenheit wurde bei Bedarf auch die Kapitelstruktur geändert oder die Verantwortlichkeit unter den Autoren neu aufgeteilt.

Die Kollaboration unter den Autoren hat gut funktioniert, trotzdem muss im Rückblick der gesamte Workflow aus meiner Sicht kritisch betrachtet werden.

- Die regelmäßige Textsynchronisation per E-Mail und Clearing-Stelle hat unnötige Zeit gekostet.

- Der Clearing-Stelle sind bei der manuellen Synchronisation öfters Fehler unterlaufen, die unter anderem damit zu erklären sind, dass beide Autoren keinen wirklichen inhaltlichen Bezug zum Text gehabt haben.

- Es gab keinen ausgewiesenen Projektleiter, der sich für Stil oder Layout des Gesamttextes verantwortlich gezeichnet hat. Jeder Autor hat darauf vertraut, dass seine Kollegen ihren Teil schon richtig schreiben werden. Im resultierenden Gesamttext konnte man daher auch sehr gut die zum Teil sehr unterschiedlichen Schreibstile der einzelnen Autoren wiedererkennen, was in der Sache kein Nachteil (ein Angebot ist kein Prosatext) aber aus Sicht einer einheitlichen Außendarstellung doch stark zu bemängeln war.

- Gegen Ende des Schreibens kam es öfters vor, dass nicht ganz klar war, welche Version des Gesamtdokumentes denn nun die richtige Version ist. Ein Zustand, der sich oftmals erst durch ein Telefonat bzw. nach sorgfältigem Durchsuchen und Lesen der sich angesammelten E-Mail-Korrespondenz aufgelöst hat.

Insgesamt lässt sich jedoch sagen, dass zum damaligen Zeitpunkt keine Alternative zu der beschriebenen Art und Weise der Texterstellung zur Diskussion stand.

2.3.2 Kollaboration in Unternehmen

Je größer Unternehmen sind, desto größer ist der Dokumentationsaufwand innerhalb eines Unternehmens. Beispiele für umfangreiche Textpublikationen sind u.a.:

- Werbematerial (z.B. Kataloge, Broschüren)
- Produktdokumentationen (z.B. Benutzerhandbücher, Spezifikationen)
- Vorgaben durch Zertifizierungen (z.B. Prozessdokumentationen)
- Knowledge–Management (z.B. Berichte, Analysen)
- Ausführliche Angebote (z.B. Ausschreibungen, Wettbewerbe)

Es ist unmittelbar klar, dass diese Textumfänge nicht von einem oder einigen wenigen Autoren erarbeitet werden kann. Dem einzelnen fehlt neben der Zeit oft auch das Wissen (z.B. bei Produktspezifikationen) oder das Können (z.B. bei zielgruppengerechten Formulieren von Werbetexten), um geforderte Text schreiben zu können.

Eine weitere Dimension erfährt die Kollaboration in Unternehmen durch zwei weitere Aspekte:

- Bedingt durch die Globalisierung agieren Mitarbeiter größerer Unternehmen oft über den Globus verteilt in unterschiedlichen Ländern und Zeitzonen.
- Bedingt durch die zunehmende Komplexität in der Technik schließen sich immer häufiger Unternehmen zu Partnernetzwerken zusammen, um beispielsweise die Entwicklungskosten neuer Produkte untereinander zu teilen.

Beides hat unmittelbare Auswirkung auf die Komplexität von Kollaboration in Unternehmen. Dies fängt damit an, dass multinationale Unternehmen sich zunächst einmal auf eine gemeinsame Sprache (in der Regel Englisch) einigen müssen und endet nicht selten in der Frage, welche kollaborativen Software-Werkzeuge für eine effektive Zusammenarbeit am geeignetsten sind.

Auch wenn der Schwerpunkt dieser Arbeit auf das kollaborative Erstellen wissenschaftlicher Texte liegt, soll diese kleine Exkursion zeigen, dass Kollaboration bei weitem nicht auf den wissenschaftlichen Bereich beschränkt ist.

2.4 Taxonomie

[Lowry et al. 04] haben sich in ihrem Artikel „Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research“ einer große Anzahl theoretischer Arbeiten zum Thema kollaboratives Schreiben angenommen, um daraus eine möglichst allgemeingültige Taxonomie zu entwickeln. Diese wird im Folgenden vorgestellt und diskutiert.

2.4.1 Was ist kollaboratives Schreiben?

Der Begriff *kollaboratives Schreiben* ist eine direkte Übersetzung des englischen Begriffs Collaborative Writing (CW), der in [Lowry et al. 04] wie folgt definiert wird:

> *„CW is an iterative and social process that involves a team focused on a common objective that negotiates, coordinates, and communicates during the creation of a common document."*

Kollaboratives Schreiben (KS) hat das Ziel, ein Dokument gemeinsam in einer Gruppe zu verfassen. Je größer die Gruppe ist und je komplexer das Dokument sein soll, desto größer ist der Aufwand, diese Arbeit zu delegieren. Die Gruppenmitglieder können sich u.a. durch ihre Qualifikation, ihre Verfügbarkeit, ihre Motivation bzw. ihre Teamfähigkeit sehr stark voneinander unterscheiden. Es ist zum Beispiel ein Unterschied, ob sich alle Gruppenmitglieder bereits seit langem kennen oder sich vorher noch nie gesehen haben. Es ist auch ein Unterschied, ob alle Gruppenmitglieder am selben Ort zusammenarbeiten oder über Ländergrenzen und Zeitzonen hinweg verteilt sind. KS ist ein komplexer Prozess, bei dem das eigentliche Schreiben des Textes nur einen Teilaspekt ausmacht. Vor, nach und während des Schreibens müssen sich alle Gruppenmitglieder stets untereinander abstimmen und synchronisieren.

2.4.2 KS-Strategien

Die technisch gesehen einfachste Möglichkeit, einen Text gemeinsam zu verfassen, ist es, ein Gruppenmitglied als alleinigen Autor zu bestimmen, der den gesamten Text schreibt. Alle anderen Gruppenmitglieder unterstützen ihn dabei (z.B. durch Informationsbeschaffung, als Diskussionspartner, etc.).

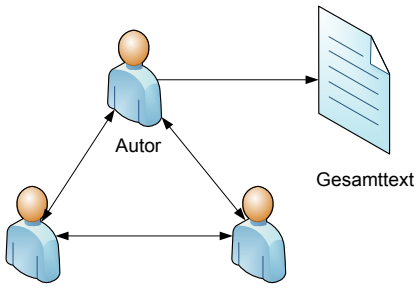

Abbildung 1: Texterstellung durch Einzelautor

Der Vorteil liegt in der effizienten Umsetzung (keine Textsynchronisation innerhalb der Gruppe ist nötig) und in der resultierenden Konsistenz des Textes (der Stil nur eines Autors). Mit zunehmendem Umfang des Textes ist diese spezielle Form von KS schon aus Zeitgründen nicht mehr durchführbar. Zudem besteht die Gefahr, dass der Autor nicht wirklich den gemeinsamen Konsens in der Gruppe in seinem Text reflektiert.

Bei einer *sequenziellen Texterstellung* fängt ein Autor an zuschreiben und gibt seinen Text zu einem späteren Zeitpunkt weiter an den nächsten Autor. Die Vorgehensweise wird so lange wiederholt, bis jeder Autor der Gruppe seinen Teiltext geschrieben hat.

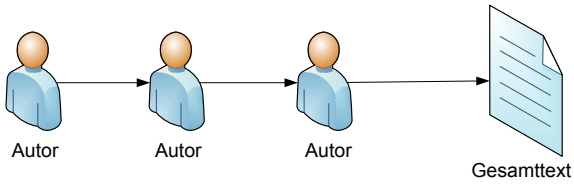

Abbildung 2: Sequentielle Texterstellung

Der Vorteil dieser Strategie liegt ebenfalls in einer einfachen organisatorischen Umsetzung, zudem kann die Texterstellung auf beliebig viele Autoren verteilt werden. Der Nachteil ist in der nicht vorhandenen Parallelität des Schreibens zu sehen. Damit dauert die Erstellung eines Textes theoretisch genauso lange wie die Erstellung durch einen gemeinsam bestimmten Einzelautor. Hinzu kommen noch eine Reihe weiterer Probleme:

- Autoren können nach Weitergabe des Textes an den ihnen folgenden Autor nicht mehr auf Meinungsänderungen oder neue Fakten reagieren.
- Jeder Autor kann die Teiltexte seiner Vorgänger verändern oder löschen, wenn dies nicht durch technische Maßnahmen unterbunden wird.
- Ein einzelner Autor kann durch unkooperatives Verhalten den gesamten Texterstellungsprozess aufhalten oder sogar ganz zum Erliegen bringen.
- Der Autor des letzten Teiltextes wird sehr stark durch die bereits vorhandenen Textteile beeinflusst, während der Autor des ersten Textteils frei von derartigen Einflüssen ist.

Bei der *parallelen Texterstellung* schreiben mehrere Autoren zeitgleich am Gesamttext. In der Regel wird dazu der Text in voneinander unabhängige Teiltexte aufgeteilt, die von jedem Autor eigenverantwortlich erstellt werden. Ein ausgewiesenes Teammitglied spielt als Koordinator die Teiltexte in ein Gesamtdokument zusammen (diese Funktion lässt sich theoretisch auch automatisieren).

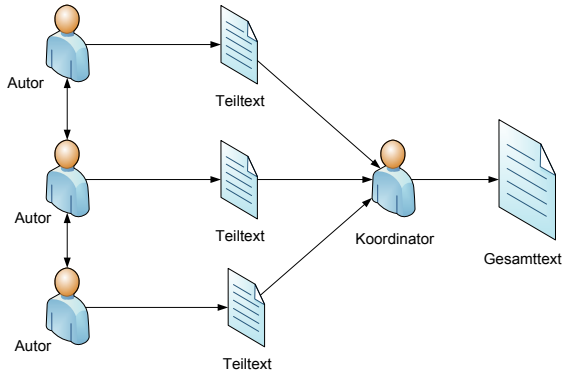

Abbildung 3: Parallele Texterstellung durch Textaufteilung

Der Vorteil dieser Strategie liegt in der erheblichen Zeitersparnis, da mehrere Autoren gleichzeitig am Text arbeiten. Zudem könne neue Erkenntnisse während des Schreibens den anderen Autoren mitgeteilt werden, so dass sie eventuell darauf in ihren Texten reagieren können.

Die Nachteile liegen in der Schwierigkeit, einen Text unter den Autoren adäquat aufzuteilen. Es ist nicht immer klar, wo genau die Grenzen zwischen den Teiltexten zu ziehen sind. Auch werden die unterschiedlichen Kompetenzen, Fähigkeiten oder Zeitkonten kaum berücksichtigt. Zudem besteht die Gefahr redundanter Texterstellung, wenn sich Autoren nicht ausreichend untereinander absprechen.

Eine Möglichkeit, diese Probleme zu minimieren, ist es, die unterschiedlichen Fähigkeiten einzelner Gruppenmitglieder mehr in Betracht zu ziehen und eine Rollenaufteilung anstatt einer Textaufteilung vorzunehmen. So könnte ein Teammitglied den eigentlichen Text schreiben, ein anderes Mitglied für Grafiken, Formeln und Tabellen verantwortlich sein, weitere Mitglieder könnten die Rolle des Editors, Layouters oder Gutachters übernehmen.

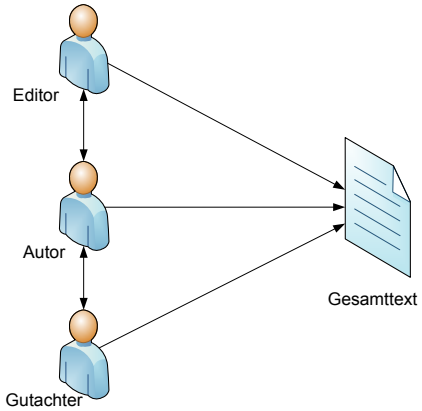

Abbildung 4: Parallele Texterstellung durch Rollenverteilung

Der Vorteil der *rollenbasierten parallelen Texterstellung* liegt in einem wesentlich flexibleren Gesamtprozess, der die unterschiedlichen Fähigkeiten der Teammitglieder berücksichtigt, während alle ihre Arbeit mehr oder weniger gleichzeitig ausfüh-

ren können. Gleichzeitig wird auch die Kommunikation unter den Mitgliedern gefördert, weil alle sehr viel mehr voneinander abhängig sind.

Der Nachteil ist, dass eine solche Strategie nicht mehr mit herkömmlichen Textverarbeitungssystemen realisiert werden kann. Es wird vielmehr ein System benötigt, das eine adäquate Textsynchronisation unter allen Beteiligten ermöglicht.

Bei einer *reaktiven Texterstellung* gibt es keine unmittelbaren organisatorischen Vorgaben für das gemeinsame Schreiben eines Textes. Alle Autoren haben vielmehr unmittelbar Zugriff auf alle Texte ihrer Mitautoren und können unmittelbar darauf reagieren und ihre eigenen Texte schreiben.

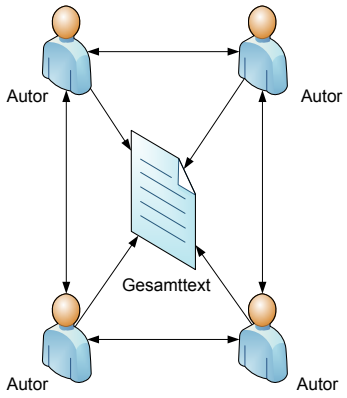

Abbildung 5: Reaktive Texterstellung

Der Vorteil liegt in einer kreativen Schreibform ohne organisatorische Restriktionen. Der unmittelbare Zugriff auf alle Texte erhöht die Kommunikationsbereitschaft. Die Autoren können somit sehr schnell auf Konsensänderungen in der Gruppe reagieren und bekommen unmittelbar Rückmeldung zu ihren eigenen Texten.

Der Vorteil kann auch gleichzeitig zum Nachteil werden. Ohne organisatorischen Rahmen kann es sehr schnell zu sozialen Spannungen in der Gruppe kommen, wenn beispielsweise ein Autor ungefragt den Text eines anderen Autors ändert oder löscht. Zudem ist diese Strategie nur unter gemeinsamer Nutzung eines für diesen Zweck spezialisierten Software-Systems möglich.

2.4.3 KS-Aktivitäten

Ein Text wird in der Regel nicht einfach nur herunter geschrieben. Das eigentliche Schreiben ist nur eine von vielen begleitenden Aktivitäten, die innerhalb eines KS-Prozesses stattfinden. Typische Aktivitäten sind in der folgende Tabelle aufgleistet:

Phase	Erklärung
Ideen sammeln	Ideen für einen ersten Entwurf artikulieren
Ideen filtern	Die gesammelten Ideen auf ihre Verwendbarkeit hin

	überprüfen
Skizzieren	Einen Text-Entwurf erstellen
Schreiben	Den Text schreiben
Nachprüfen	Den Text auf Mängel hin überprüfen
Korrigieren	Den Text auf Grund von Rückmeldungen und neuen Erkenntnissen bearbeiten.
Editieren	Den Text in seine endgültige Endfassung bringen

Tabelle 1: Typische Aktivtäten

Diese Aktivitäten müssen nicht notwendigerweise sequentiell aufeinander folgen. So können bereits fertiggestellte Teiltexte einer Nachprüfung unterzogen werden, ohne dass der Gesamttext fertiggestellt ist. Auch sind nicht notwendigerweise alle Aktivitäten notwendig. Ist beispielsweise der Inhalt des Textes bereits vorgegeben, können die ersten beiden Aktivitäten des Ideen sammeln und filtern durchaus wegfallen.

2.4.4 KS-Textkoordination

Ein KS-Prozess muss wie jeder von Menschen ausgeführte Prozess beaufsichtigt werden. Zeitliche (z.B. Dead-Line), technische (z.B. Software-Probleme) und zum Teil auch soziale (z.B. Autor liefert keinen Input) Probleme müssen erkannt und gelöst werden. Dazu muss festgelegt werden, wer die Oberaufsicht über den Gesamttext hat.

Bei einer *zentralen Textkoordination* ist ein ausgewiesenes Gruppenmitglied verantwortlich für den Gesamttext. Diese Art der Koordination eignet sich sehr gut für alle KS-Strategien mit Ausnahme der reaktiven Texterstellung.

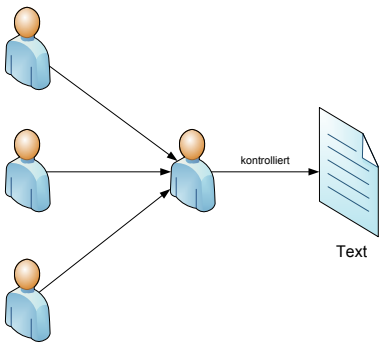

Abbildung 6: Zentrale Textkoordination

Bei einer *wechselnden Textkoordination* übernimmt zunächst ein Teammitglied die zentrale Koordination für den Gesamttext, gibt diese aber nach einer bestimmten Zeit an ein anderes Teammitglied ab. Diese Art der Koordination eignet sich sehr gut für die sequentielle Texterstellung.

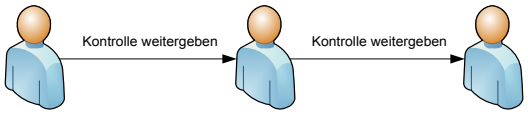

Abbildung 7: Wechselnde Textkoordination

Bei einer *unabhängigen Textkoordination* koordiniert jedes Teammitglied einen disjunkten Teiltext. Diese Art der Koordination eignet sich sehr gut für die parallele Texterstellung durch Textaufteilung.

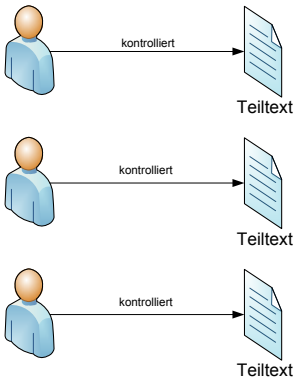

Abbildung 8: Unabhängige Textkoordination

Bei einer *verteilten Textkoordination* koordinieren alle Teammitglieder gleichberechtigt den Gesamttext. Diese Art der Koordination eignet sich sehr gut für die reaktive Texterstellung.

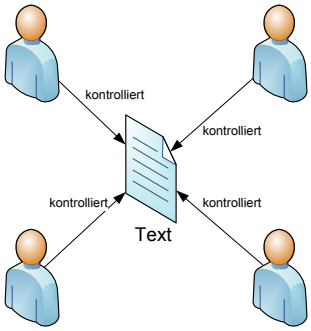

Abbildung 9: Verteilte Textkontrolle

2.4.5 KS-Rollen

In einem KS-Prozess nehmen Gruppenmitglieder unterschiedliche Rollen ein. Typische Rollen sind in der folgenden Tabelle aufgelistet:

Rolle	Beschreibung
Autor	Ein Teammitglied, das für den Inhalt eines Teiltextes verantwortlich ist.
Berater	Ein Teammitglied, das andere Teammitglieder mit Informationen, Analysen und Expertisen unterstützt.
Editor	Ein Teammitglied, das für den Inhalt und die endgültige Version des Gesamttextes verantwortlich ist.
Gutachter	Ein Teammitglied, das qualifizierte Rückmeldung zur Textinhalten gibt.
Teamleiter	Ein ausgewiesenes Teammitglied, das alle anderen Teammitglieder durch den gesamten KS-Prozess führt.
Lektor	Eine Person außerhalb des Teams, die Form, Stil und Inhaltsauswahl des Gesamttextes vorgibt und auf die Einhaltung dieser Vorgaben durch die Autoren achtet. Inwieweit sie auch den Inhalt der Texte selbst beurteilen kann wie dies im literarischen Bereich z.B. der Fall ist, hängt bei wissenschaftlichen Texten von der Qualifikation des Lektors ab, kann aber hier vernachlässigt werden, da diese Rolle dem Editor zufällt.

Tabelle 2: Typische Rollen

Diese Liste lässt sich natürlich beliebig erweitern. Die möglichen Rollen, welche die Gruppenmitglieder einnehmen können, hängen unter anderem ab von der gewählten KS-Strategie, der Größe und der Zusammensetzung der Gruppe sowie der Komplexität des Textes.

2.4.6 KS-Arbeitsmodi

Der Arbeitsmodus innerhalb eines KS-Prozesses wird durch zwei Parameter bestimmt:

1. die physikalische Nähe und
2. die Synchronität der Arbeit.

Es ist ein Unterschied, ob sich alle die Gruppenmitglieder im selben Raum aufhalten oder sich über das Gebäude, über verschiedene Städte oder Länder verteilen. Es ist

auch ein Unterschied, ob alle Gruppenmitglieder gleichzeitig (synchron) oder zeitlich versetzt (asynchron) arbeiten.

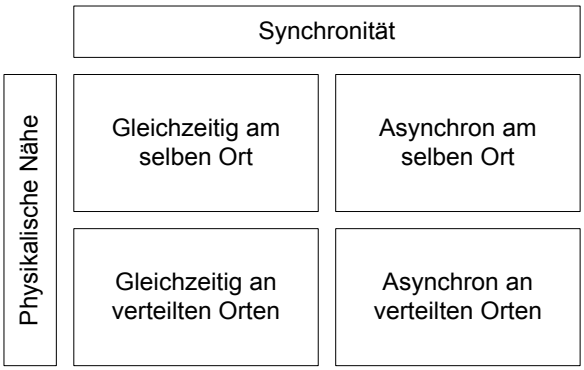

Abbildung 10: Arbeitsmodi

Sind alle Gruppenmitglieder zur selben Zeit am selben Ort, gestaltet sich die Kommunikation am einfachsten (miteinander sprechen), allerdings ist die Koordinierung am restriktivsten (nicht jeder kann immer zur selben Zeit am gleichen Ort sein, insbesondere dann, wenn Autoren in verschiedene Zeitzonen arbeiten). Können Gruppenmitglieder zur selben Zeit an unterschiedlichen Orten arbeiten, erleichtert dies die Koordinierung, gleichzeitig aber muss die Kommunikation durch Telefon oder Chat ersetzt werden. Bei einer asynchronen Zusammenarbeit verlagert sich auch die Kommunikation auf asynchrone Werkzeuge wie beispielsweise E-Mail. Dabei ist es im Prinzip unerheblich, ob alle Gruppenmitglieder am selben oder an unterschiedlichen Orten arbeiten.

2.4.7 Diskussion

Eine ganz ähnliche Taxonomie mit fast den gleichen Begriffen wurde schon 1992 in einer älteren Studie aufgestellt [Posner et al. 92]. Es soll nicht unerwähnt bleiben, dass jene Studie und damit indirekt auch die hier vorgestellte Taxonomie durchaus kritisch gesehen werden kann. In [Noël et al. 04] wird beispielsweise darauf hingewiesen, dass die Grundlage der älteren Studie auf nur wenigen Interviewteilnehmern basiert (es waren zehn) und somit die Allgemeingültigkeit der herausgearbeiteten Charakteristiken in Frage gestellt werden muss. Auf der anderen Seite spricht die Tatsache, dass diese Studie auch mehr als 10 Jahre nach ihrem Erscheinen immer noch eine zentrale Inspiration neben all den parallel herangezogenen Veröffentlichungen darstellt, durchaus für ihre Qualität. Tatsache ist auch, dass die meisten anderen Studien und Artikel gar nicht erst den Versuch unternehmen, ein alternatives Vokabular zu entwickeln.

Natürlich ist unmittelbar klar, dass [Lowry et al. 04] keinen Anspruch auf Vollständigkeit oder Allgemeingültigkeit erheben können. So sind beispielsweise die einzelnen KS-Strategien nicht exklusiv zu betrachten. Sie können im Laufe eines KS-Prozesses wechseln, und durch geschickte Verknüpfung zweier Strategien ergeben sich wiederum weitere, wesentlich komplexere Strategien. Auch die aufgelisteten KS-Rollen und KS-Phasen sind beliebig erweiterbar.

Trotzdem, diese vorgestellte Taxonomie besitzt zwei Vorteile: Zum einen erleichtert die begriffliche Definition der unterschiedlichen KS-Aspekten die weitere Diskussion. Zum anderen entsteht ein möglicher Referenzrahmen für die Analyse von KS-Software-Werkzeugen. Beispielsweise können derartige Werkzeuge dahingehend untersucht werden, ob sie vollständig oder nur teilweise die definierten KS-Strategien, KS-Phasen, KS-Rollen oder KS-Arbeitsmodi unterstürzen.

2.5 Schlussbetrachtung

Wissenschaftler aus unterschiedlichen Ländern arbeiten heute ganz selbstverständlich an gemeinsamen Forschungsprojekten. Multinationale Unternehmen haben ihre Forschungs- und Entwicklungsabteilungen rund um den Globus verteilt. Ort oder Zeit sind in der Regel keine entscheidenden Faktoren mehr für ein gemeinsames Kollaborationsvorhaben, lediglich die Fähigkeiten der einzelnen Kollaborationsteilnehmer. Diese Asynchronität von Raum und Zeit (siehe KS-Arbeitsmodi) scheint mir in diesem Zusammenhang das größte Problem darzustellen. Wie stelle ich sicher, dass zwei Menschen ihre Texte miteinander synchronisieren können, obwohl sie nie oder selten zur gleichen Zeit oder am selben Ort daran arbeiten? Das nächste Kapitel versucht eine Antwort darauf zu finden, in dem es einen Blick auf die vorhandenen Software-Werkzeuge zur Textkollaboration wirft.

Darüber hinaus scheint mir jene Asynchronität noch zwei weitere Dimensionen zu besitzen: Eine soziale und eine kulturelle. Wenn zwei oder mehr Menschen zusammenarbeiten, haben diese in der Regel unterschiedliche Vorstellung, wie das gemeinsame Projekt geplant und ausgeführt werden soll. Im Idealfall kommuniziert man miteinander und einigt sich auf die notwendigen Gemeinsamkeiten. Dies ist jedoch nicht immer der Fall. Neben offenen Streit kann es noch eine Vielzahl von unausgesprochenen Konflikten geben, die einen unmittelbaren (meist negativen) Einfluss auf eine Kollaboration haben können. Beispiele hierfür sind persönliche Probleme (Gesundheit, Unlust, Überarbeitung), Neid (Verantwortung, Bezahlung) oder Überforderung (technisch, inhaltlich). Dies ist der soziale Aspekt.

Wenn eine Kollaboration über Sprach- und Kulturgrenzen hinausgeht, kann noch ein kultureller Aspekt hinzukommen. Unterschiedliche Sprachen, Traditionen und Ausbildungen können sehr schnell zu Missverständnissen führen, die eine Kollaboration in ihrer Effizienz beeinflussen können. Wenn beispielsweise die sprachliche Kommunikation nicht mehr in der eigenen Muttersprache geführt werden kann oder gar mit Hilfe eines Übersetzers geführt werden muss, können viele Informationen verloren gehen. Und gerade in einer Textkollaboration ist die Sprache doch das entscheidende Ausdrucksmittel.

Beide Aspekte sollen nicht weiter in dieser Arbeit thematisiert werden, werden aber auch nicht gänzlich ausgeblendet. So kann beispielsweise der aus Neid entstehende Drang nach Sabotage durch organisatorische Maßnahmen wie eindeutige Identifizierung, eingeschränktem Zugriff oder Änderungsprotokollierung weitestgehend neutralisiert werden.

3 KS-Software

3.1 Einführung

Der Computer ist heute das dominierende Werkzeug zum Verfassen von Texten jeglicher Art. Mit der Entwicklung der Hardware ging auch die Entwicklung der Software, sogenannter Textverarbeitungssysteme, einher. Waren es am Anfang noch einfache Texteditoren, welche lediglich die Eingabe und das Zusammenfügen von Buchstaben zu Wörtern, Sätzen und Texten ermöglichten, bieten heutige Software-Lösungen umfangreiche Möglichkeit, Texte grafisch ansprechend zu gestalten und mit Medienformaten wie Grafiken und Fotos anzureichen. Aus diesem Grund hat sich das Bild der automatisierten Textverarbeitung über die Jahre geändert, so dass zunächst eine Präzisierung dieses zentralen Begriffes angebracht erscheint, die man wie folgt vornehmen könnte:

> *Ein Textverarbeitungssystem ist eine Software zum Erstellen, Bearbeiten und Organisieren von Texten jeglicher Art. Dies schließt auch die grafische Aufbereitung durch Textformatierung und das Verknüpfen mit Medienformaten wie Grafiken, Fotos, Animationen und Videos mit ein.*

Die heutige Vielfallt von Textverarbeitungssystemen bedarf zunächst einer Systematisierung, die im Folgenden vorgenommen werden soll. Ein möglicher Ansatz ist es, Textverarbeitungssysteme auf Grund ihrer Kernfunktionalität zu klassifizieren:

- **Texteditoren** – Texteditoren ermöglichen das Erstellen und Editieren eines Textes unter Negierung jeglicher Metadaten (wie z.B. grafischen oder logischen Formatierungsanweisungen). Die elementare Einheit, der einzelne Buchstabe bzw. das einzelne Zeichen, gruppiert zu Wörtern ist zugleich auch die höchste Form der Abstraktion. Texteditoren reduzieren das Schreiben von Texten auf die reine Zeicheneingabe. Ein bekanntes Beispiel dafür ist die Anwendung „Editor“ (englisch: „Notepad“) unter den heute gängigen Windows-Betriebssystemen.[1]

- **Textsatzsysteme** – Ein Textsatzsystem ist eine Software, die einen Text automatisiert in ein gewünschtes Ausgabeformat überführt. Anstatt konkrete Aussagen über das endgültige Layout zu machen (Schriftart, Schriftgröße, Schriftstil etc.) wird der Text logisch formatiert (Überschrift, Zitat, Formel, etc.). Dadurch entsteht eine höhere Abstraktionsebene, in der strikt zwischen Inhalt und Layout unterschieden wird. Bekannte Beispiele für Textsatzsysteme sind LaTeX (www.latex-project.org) und TUSTEP (www.tustep.uni-tuebingen.de).

[1] Da die Anwendung „Editor“ eine doch recht rudimentäre Funktionalität aufweist, gibt es eine nahezu unüberschaubare Anzahl an Alternativen, die das Grundkonzept neu interpretieren (z.B. Notepad++, Notepad2, usw.).

- **WYSIWYG-Textsysteme** – Die Abkürzung WYSIWYG steht für das englische „What You See Is What You Get“, welches Textverarbeitungssysteme kennzeichnet, bei denen der Autor bereits während der Texterstellung auf dem Bildschirm das grafische Endergebnis sieht. Änderungen jeglicher Art (inhaltlich wie gestalterisch) werden unmittelbar umgesetzt. Wird beispielsweise der Zeilenabstand innerhalb eines Absatzes verändert, so lässt sich das Ergebnis unmittelbar auf dem Bildschirm nachvollziehen. Bei WYSIWYG-Systemen wird oft noch zwischen WYSIWYG-Textverarbeitungssystemen und Desktop-Publishing-Systemen (DTP-Systemen) unterscheiden. DTP-Systeme unterscheiden sich von Textverarbeitungssystemen dahingehend, dass sie ihren Schwerpunkt eher in der punktgenauen grafischen Gestaltung eines Textes in Verbindung mit Medienformaten wie Grafiken und Fotos als in der effizienten Organisation und Verarbeitung großer Texte sehen. Beispiele für WYSIWYG-Textverarbeitungssysteme wären Microsoft Word, Open Office Writer und Google Text. Bekannte Beispiele für DTP-Systeme sind Adobe InDesign und QuarkExpress.

Die Abgrenzung zwischen den Textverarbeitungssystemen kann in heutiger Zeit als fließend betrachten werden. So sind beispielsweise Satzsysteme in der Regel auf das Vorhandensein von Texteditoren angewiesen, da sie oft keine eigene bzw. eine nur sehr unzureichende Editierkomponente besitzen. Auch besitzen moderne WYSWIG-Textverarbeitungssysteme genug Funktionalität, um Inhalt und Layout sauber zu trennen. Umgekehrt existieren mittlerweile Aufsätze für Satzsysteme, die dem Autor das Gefühl einer WYSIWYG-Textverarbeitung vermitteln.

Diese traditionelle Klassifizierung von Textverarbeitungssystemen hat in den letzten Jahren eine zusätzliche Dimension erhalten. Unter dem Stichwort „Web 2.0“ hat sich das Internet von einem reinen Informationsmedium hin zu einem Kollaborationsmedium weiterentwickelt. Diese Entwicklung hat auch bei der Textverarbeitung nicht halt gemacht, so dass mir die folgende zusätzliche Klassifizierung sinnvoll erscheint:

- **Offline-Textverarbeitung** – Offline bedeutet, dass ein Computer keine aktive Netzwerkverbindung zu anderen Computern besitzt, somit also keine Kommunikation mit anderen Computern möglich ist. Eine Textverarbeitung, die unter diesen Rahmenbedingungen operieren kann ist eine Offline-Textverarbeitung. Beispiele hierfür wären Microsoft Word oder Open Office Writer.

- **Online-Textverarbeitung** – Online bedeutet, dass ein Computer eine oder mehrere aktive Netzwerkverbindungen zu anderen Computern besitzt. Die Art des Netzwerks kann dabei von einem einfachen lokalen Netzwerk, bei dem zwei oder mehrere Computer zusammengeschaltet werden, bis hin zu einer direkten Anbindung an das Internet reichen. Entscheidend dabei ist, dass die aktive Netzwerkverbindung Voraussetzung für den Einsatz der Textverarbeitung ist. Beispiele für Online- Textverarbeitungen wären Google Text und Microsoft Web Word.

Auch hier ergibt eine nähere Betrachtung, dass diese Abgrenzung oft rein theoretischer Natur ist. So muss zunächst unterscheiden werden, ob die gesamte Anwendung als Online-Dienst zur Verfügung steht (z.B. Google Text) oder lediglich der Text online gehalten wird (z.B. Microsoft Word 2010). Zudem besitzen manche Textverarbeitungssysteme die Fähigkeit zwischen einem Online- und Offline-Modus wechseln zu können (z.B. Offline-Funktionalität von Google Text mit Hilfe von Google Gears). Der entscheidende Punkt bei der Betrachtung von Textverarbeitungssystemen jedoch ist, dass deren Online-Fähigkeit die Möglichkeit zur Kollaboration unter Einzelautoren eröffnet. Und genau diese software-technische Unterstützung des kollaborativen Schreibens soll Thema der folgenden Kapitel sein.

Kollaboratives Schreiben impliziert die Verfügbarkeit von passenden Textverarbeitungssystemen. Der Frage, wie genau so eine Software auszusehen hat und was sie leisten soll, soll sich zunächst mit einem Überblick über aktuelle Textverarbeitungssysteme und deren kollaborativen Fähigkeiten genähert werden. Anschließend wird ein Anforderungskatalog für kollaborative Textverarbeitungssysteme erstellt mit dessen Hilfe die beiden meiner Meinung nach interessantesten Vertreter der zuvor vorgestellten Textverarbeitungssysteme analysiert werden sollen.

3.2 Kollaborative Textverarbeitungssysteme

3.2.1 Klassische Textverarbeitung

Gegenwärtig sind drei Software-Pakete für alle zugänglich und daher entsprechend weit verbreitet. Sie alle besitzen eine Textverarbeitung mit kollaborativen Fähigkeiten, die im Folgenden vorgestellt werden:

- **Microsoft Word 2010** ist ein umfangreiches kommerzielles Software-Paket der Firma Microsoft für die Betriebssysteme Microsoft Windows und Apple Mac OS X. Es besteht u.a. aus den Einzelanwendungen *Word* (Textverarbeitung), *Powerpoint* (Präsentation) und *Excel* (Tabellenkalkulation).

- **iWorks 09** ist ein kommerzielles Software-Paket der Firma Apple für das Betriebssystem Apple Mac OS X. Es besteht aus den Einzelanwendungen *Pages* (Textverarbeitung), *Keynote* (Präsentation) und *Numbers* (Tabellenkalkulation).

- **OpenOffice.org Version 3** ist ein frei verfügbares Open-Source-Software-Paket für die Betriebssysteme Microsoft Windows, Apple Mac OS X, Linux und zahlreiche Unix-Varianten. Es besteht aus den Modulen *Writer* (Textverarbeitung), *Math* (Formeln), *Calc* (Tabellenkalkulation), *Draw* (Vektorgrafik), *Impress* (Präsentation) und *Base* (Datenbankoberfläche).

Word, Pages und Writer erlauben alle drei die Gliederung eines Dokuments in ein Zentraldokument und in beliebig viele Filialdokumente. Ein Zentraldokument ist ein Container für eine bestimmte Anzahl voneinander getrennter Dateien, mit dem ein

aus mehreren Teilen bestehendes Dokument (z.B. ein Buch mit mehreren Kapiteln) erstellt und verwaltet werden kann.

Word, Pages und Writer erlauben es, Änderungen an einem Text mit zu protokollieren. Damit kann jedes Gruppenmitglied jederzeit gelöschte, eingefügte oder anderweitig bearbeitete Textstellen sichtbar machen. Geänderte Texte und Grafiken werden durch farbliche Markierungen hervorgehoben. Das Gruppenmitglied kann entscheiden, ob er die Änderungen übernimmt oder wieder verwirft. Werden Änderungen von mehreren unterschiedlichen Gruppenmitgliedern getätigt, werden diese durch Farbe und Namenskürzel unterschieden.

Word, Pages und Writer erlauben es, einen Text mit Kommentaren zu versehen. Damit kann ein Gruppenmitglied den Text eines anderen Gruppenmitglieds mit Notizen und Anmerkungen versehen. Diese werden in einer Sprechblase am Rand des Dokuments oder in einem Überarbeitungsfenster angezeigt. Werden Kommentare von mehreren unterschiedlichen Gruppenmitgliedern eingefügt, werden diese durch Farbe und Namenskürzel unterschieden.

Word, Pages und Writer ermöglichen es, das aktuelle Dokument direkt per E-Mail an andere Gruppenmitglieder zu versenden.

Word erlaubt einen gemeinsamen Zugriff auf Dokument über das Internetportal Office Live Workspace. Grundlage für den Austausch von Dokumenten sind sogenannte Arbeitsbereiche. Jeder Arbeitsbereich besitzt einen Namen und kann beliebig viele Dokumente enthalten. Innerhalb eines Arbeitsbereichs können Dokumente wiederum hierarchisch in Ordnern organisiert werden. Der Besitzer eines Arbeitsbereichs hat nun die Möglichkeit den gesamten Arbeitsbereich, einzelne Dokumente oder seinen aktuellen Bildschirm anderen Gruppenmitgliedern freizugeben.

Bei der Freigabe wird zwischen „Editoren“ und „Anzeigende Benutzer“ unterschieden. „Editoren“ erhalten vollen Lese- und Schreibzugriff, können Inhalte löschen und mit Kommentaren versehen. „Anzeigende Benutzer“ erhalten lediglich Lesezugriff und können Kommentare verfassen.

Alle Aktivitäten (wer hat was und wann in einem Arbeitsbereich erstellt, geändert oder gelöscht) können jederzeit nachvollzogen werden. Auf Wunsch kann ein Gruppenmitglied bei jeder neuen Aktivität per E-Mail benachrichtigt werden.

iWorks erlaubt den gemeinsamen Zugriff auf Dokumente über das Internetportal iworks.com. Vor der Veröffentlichung des Dokuments wird festgelegt, welche Gruppenmitglieder Zugriff erhalten sollen und in welcher Form sie Zugriff erhalten sollen. Standardmäßig kann das Dokument nur im Internet-Browser betrachtet werden. Alle Gruppenmitglieder werden automatisch per E-Mail darüber benachrichtigt, dass ein neues Dokument bereitsteht. Gruppenmitglieder können zusätzlich beliebige Notizen hinterlegen. Je nach erteilter Berechtigung kann das Dokument auch auf den eigenen Rechner heruntergeladen und verändert oder mit Kommentaren versehen werden. Beim Herunterladen des Dokuments kann zusätzlich zwischen den Formaten PDF, DOC oder RTF gewählt werden.

Dokumente können in Pages und Writer nicht parallel bearbeiten werden. Hat ein Gruppenmitglied beispielsweise ein Dokument zur Bearbeitung geöffnet und ein weiteres Gruppenmitglied möchte diese Datei ebenfalls aufrufen, erhält dieses einen Warnhinweis und kann die betreffende Datei wahlweise nur mit Leserechten öffnen oder einen Hinweis erhalten, sobald die Datei wieder zur Verfügung steht

Word erlaubt es dagegen mit zwei oder mehreren Benutzern auf einem gemeinsamen Textdokument zu arbeiten. Dies setzt jedoch eine SharePoint-Infrastruktur voraus. SharePoint ist eine Server-Plattform zur Bereitstellen von Kollaborations- und Web-Publishing-Diensten der Firma Microsoft. Grundsätzlich gibt es zwei Möglichkeiten, SharePoint nutzbar zu machen:

1. Installation von SharePoint 2010 auf einem Server im Netzwerk oder Internet.

2. Nutzung von SkyDrive, einem Internetdienst von Microsoft, der eine virtuelle Festplatte im Internet zur Verfügung stellt. Da SkyDrive auf der SharePoint-Infrastruktur aufbaut, kann dieser Dienst ebenfalls als Kollaborationsplattform genutzt werden. Die Nutzung von SkyDrive setzt voraus, dass alle Benutzer eine Windows-Live-ID[2] besitzen.

Möchten zwei oder mehr Benutzer gemeinsam an einem Dokument arbeiten, müssen sie zunächst wie folgt vorgehen:

1. Ein Benutzer muss auf dem SharePoint-Server oder unter SkyDrive ein neues Dokument anlegen. Dies geschieht, in dem er ein neues oder bereits vorhandenes lokales Dokument unter der gewünschten URL abspeichert.

2. Anschließend fügt dieser Benutzer weitere Benutzer als Berechtigte zu diesem Dokument hinzu. Er hat dabei die Wahl zwischen den Berechtigungen „Ansicht möglich“ und „Bearbeitung möglich“.

Von diesem Zeitpunkt an können alle berechtigten Benutzer gemeinsam auf diesem Dokument arbeiten. Dabei stehen allen Benutzern folgende Möglichkeiten zur Verfügung.

- Jeder Benutzer sieht alle anderen aktiven Benutzer. Für jeden Benutzer werden sein Name und seine Kontaktdaten angezeigt. Weitere Optionen sind:
    - Unmittelbares Versenden einer E-Mail
    - Unmittelbares Starten einer Chat-Kommunikation
    - Unmittelbares Starten eines Telefonats

[2] Das Anlegen einer Windows-Live-ID ist kostenlos und ermöglicht den Zugriff auf eine Reihe kostenloser wie kostenpflichtiger Dienste von Microsoft. SkyDrive ist ein kostenloser Dienst.

    - Unmittelbare Planung eines Besprechungstermins

- Editiert ein Benutzer einen Absatz, wird dieser für andere Benutzer solange geblockt, bis dieser Absatz gespeichert wurde.

- Soll einem bestimmten Benutzer dauerhaft die Bearbeitung eines ganzen Abschnitts (z.B. eines Kapitels) nicht erlaubt werden, so kann dieser Benutzer geblockt werden.

- Sobald ein Benutzer seine lokalen Änderungen speichert, werden diese auf den SharePoint-Server übertragen und in das zentrale Dokument eingespielt. Zuvor wird die vorherige Version des Dokuments gespeichert. Gleichzeitig werden die gespeicherten Änderungen aller anderen Benutzer lokal geladen.

- Tritt beim Speichern der lokalen Änderungen ein Konflikt auf, so wird der Benutzer drauf hingewiesen. Er hat nun die Möglichkeit, diese Konflikte manuell aufzulösen. Dazu werden ihm alle Konflikte in einer Liste angezeigt. Der Benutzer hat nun für jeden Konflikt die Wahl zwischen „Annehmen“ oder „Ablehnen“ der fremden Änderung. Erst wenn alle Konflikte manuell aufgelöst sind, kann der Benutzer im Dokument weiterarbeiten.

- Über die Änderungsverfolgung können die Änderungen anderer Benutzer grafisch in das aktuelle Dokument integriert und damit nachvollzogen werden.

- Über den Dokumentenvergleich kann die aktuelle Version des Dokuments mit älteren Versionen, die automatisch auf dem SharePoint Server angelegt werden, verglichen werden.

- Einzelne Wörter, Sätze oder Absätze können mit einem Kommentar versehen werden, den alle anderen Benutzer lesen können.

- Einschränkungen: Der maximale Speicherplatz auf SkyDrive ist zur Zeit auf 25 GB beschränkt.

Die Speicherplatzbeschränkung unter SkyDrive ist großzügig bemessen und dürfte für viele Dokumentenumfänge ausreichen. Lediglich bei einer größeren Anzahl an Dokumenten oder bei Dokumenten, die außer Text intensiv weitere speicherintensive Quellen wie z.B. Videos einbinden, können diesen Speicherplatzrahmen schnell überschreiten. In diesen Fällen wäre ein Ausweichen auf eine selbst-administrierte SharePoint-Infrastruktur anzuraten.

3.2.2 Dokumenten-Management-Systeme

Dokumenten-Management-Systeme (DMS) sind datenbankgestützte Softwaresysteme zur Verwaltung von elektronischen Dokumenten. Man unterscheidet dabei zwischen Dokumenten-Management im engeren Sinne und Dokumenten-Management im weiteren Sinne [Kampffmeyer et al. 99].

Dokumenten-Management im engeren Sinne umfasst klassische Software-Systeme, die sich ausschließlich auf die Verwaltung von elektronischen Dokumenten beschränken. Typische Funktionen solcher Systeme sind:

- Metadatenverwaltung zur Beschreibung und Indizierung von Dokumenten
- Aufbau und Visualisierung von Organisationsstrukturen
- Umfangreiche Such- und Recherchemöglichkeiten
- Versionsverwaltung von Dokumenteninhalten
- Zugriffsmechanismen zum Schutz vor unerlaubten Zugriff
- Sicherungsmechanismen zum Schutz vor Datenverlust

Dokumenten-Management im weiteren Sinne umfasst zusätzlich ein weites Spektrum von Technologien, die sich im Laufe der Zeit um die klassische Funktionalität herum gruppiert haben. Dazu zählen beispielsweise Workflow-Management, Langzeitarchivierung, Scanner-Technologien und automatische Texterkennung.

Damit Software-Anwendungen ihre Dokumente in einem DMS ablegen bzw. wieder auslesen können, benötigen sie klar definierte Zugriffsschnittstellen. Da die Software-Hersteller jedoch nicht für jedes DMS eine extra Anbindung entwickeln wollen, wurden im Lauf der Jahre einige Technologien bzw. Standards entwickelt, die einen transparenten Zugriff erlauben sollen:

- ***Virtuelles Dateisystem:*** Eine Möglichkeit, den Zugriff auf ein DMS vollkommen transparent zu gestalten, ist es, alle Dokumente durch ein virtuelles Dateisystem zu publizieren. Ein virtuelles Dateisystem ist Dateisystem, dessen Inhalt (Dateien und Ordner) sich durch Zugriff auf einen physikalischen Datenträgers (z.B. Festplatte) speist, sondern durch Zugriff auf einen Software-Dienst, nämlich dem DMS, das sich auf einem anderen Computer im Netzwerk befindet. Der Vorteil liegt nun darin, dass Software-Anwendungen keine spezielle Anpassung machen müssen, um das DMS zu unterstützen. Das Auswählen eines Ordners oder einer Datei im Dateisystem genügt. Der Nachteil liegt in der hohen Systemnähe, d.h. für jedes Betriebssystem muss das DMS eine entsprechend Implementierung seines virtuellen Dateisystems vornehmen.

- ***ODMA:*** Die Open Document Management API (ODMA) wurde 1994 durch einen Zusammenschluss mehrerer Software-Hersteller geschaffen [ODMA 97]. Es handelt sich dabei um eine standardisierte Programmierschnittstelle für die einfache Integration von DMS in Desktop-Anwendungen. Beispielweise unterstützen einige Microsoft Office-Anwendungen (WinWord, Powerpoint) diesen Standard, so dass ein Benutzer in der Lage ist, ein Dokument direkt aus Microsoft Office heraus in das DMS zu speichern bzw. aus dem DMS heraus zu öffnen. Der ODMA-Standard gilt allerdings nicht mehr

als zeitgemäß, da derartige API-Schnittstellen in zunehmendem Maße durch protokollorientierten Service-Schnittstellen ersetzt werden.

- ***WebDAV:*** Web-based Distributed Authoring and Versioning (WebDAV) ist ein Protokollstandard der IETF (Internet Engineering Task Force) [Dusseault 04]. Mit diesem Protokoll ist es möglich, auf Basis von HTTP (Hypertext Transfer Protocol) Dateien und ganze Verzeichnisse auf einen Web-Server hoch zu laden oder von diesem herunter zu laden. Dateien können kopiert, verschoben, umbenannt, gelöscht, gesperrt und versioniert werden. WebDAV ist mittlerweile in allen modernen Betriebssystemen (z.B. Microsoft Windows, Linux, Mac OS) standardmäßig eingebaut, so dass der Benutzer über den normalen Datei-Browser auf WebDAV-kompatible Dateiablagen zugreifen kann.

Da die Technik des virtuellen Dateisystems Teil des Betriebssystems ist, können Textverarbeitungssysteme wie Word, Pages oder Writer darüber sehr leicht an ein DMS angebunden werden. ODMA wird lediglich in Word offiziell unterstützt. WebDAV wird von Word und Writer offiziell unterstützt. Pages unterstützt WebDAV indirekt durch die Integration im Apple-Betriebssystem[3].

3.2.3 Kollaborative Texteditoren

SubEthaEdit (www.subethaedit.de) ist ein kommerzieller, kollaborativer Texteditor einer Gruppe von Studenten der Technischen Universität München für das Betriebssystem Mac OS X. *SubEthaEdit* erlaubt das gleichzeitige Bearbeiten eines Textes in Echtzeit, ohne dass bestimmte Teile des Textes gesperrt werden müssen. Die Änderungen anderer Autoren werden in der eigenen Instanz farblich hervorgehoben. SubEthaEdit kommt eine zentrale Server-Instanz aus und verbindet sich automatisch mit anderen Client-Instanzen im Netzwerk.

Ähnliche Texteditoren sind ACE (sourceforge.net/projects/ace), Gobby (gobby.0x539.de) und MoonEdit (moonedit.com).

3.2.4 Web-basierte kollaborative Textverarbeitung

Google Docs (docs.google.com, auf Deutsch „Google Text & Tabellen“) ist eine kostenlos nutzbare Sammlung von Web-Applikationen der Firma Google. Es besteht aus den Einzelanwendungen *Google Documents* (Textverarbeitung, auf Deutsch „Google Text“), *Google Presentations* (Präsentation) und *Google Spreadsheets* (Tabellenkalkulation).

Google Text erlaubt es, mit zwei oder mehreren Benutzern auf einem gemeinsamen Textdokument zu arbeiten. Google Text ist eine Browser-Anwendung, die auf der Internet-Infrastruktur der Firma Google basiert. Voraussetzung für die Nutzung vom

[3] Interessanterweise basiert das Internetportal iworks.com selbst auf WebDAV.

Google Text ist ein Google-Konto[4]. Möchten zwei oder mehr Benutzer gemeinsam an einem Dokument arbeiten, müssen sie zunächst wie folgt vorgehen:

1. Ein Benutzer muss sich an seinem Google-Konto anmelden und ein neues Dokument anlegen.

2. Anschließend werden weitere Benutzer für dieses Dokument eingeladen. Jedem Benutzer muss eine der folgenden Rollen zugewiesen werden:

    - Besitzer: Erlaubt das Editieren und Löschen von Dokumenten und das Einladen von Kollaborateuren und Betrachtern.

    - Kollaborateure: Erlaubt das Editieren von Dokumenten und das Ein- und Ausladen (Berechtigung durch den Besitzer vorausgesetzt) von Kollaborateuren und Betrachtern. Es darf eine Kopie des Dokuments auf der eigenen lokalen Festplatte angelegt werden.

    - Betrachter: Erlaubt lediglich das Lesen der aktuellsten Version des Dokuments. Es darf eine Kopie des Dokuments auf der eigenen lokalen Festplatte angelegen.

Von diesem Zeitpunkt an können alle berechtigten Benutzer gemeinsam auf diesem Dokument arbeiten. Dabei stehen allen Benutzern folgende Möglichkeiten zur Verfügung.

- Dokumente können von verschiedenen Autoren zur gleichen Zeit bearbeitet, gelesen und kommentiert werden.

- Jeder Benutzer sieht alle anderen aktiven Benutzer. Für jeden Benutzer werden sein Name und seine Kontaktdaten angezeigt. Weitere Optionen sind:

    - Unmittelbares Versenden einer E-Mail

    - Unmittelbares Starten einer Chat-Kommunikation

- Alle Änderungen werden in Echtzeit unter allen Autoren synchronisiert und durch farbliche Markierungen unterschieden.

- Alle Änderungen können über eine Historie nachvollzogen werden.

- Einschränkungen: Dokumente dürfen nicht größer als 500 KB, eingebettete Bilder nicht größer als 2 MB sein. Die Anzahl der Dokumente ist auf 5000 begrenzt. Der Umfang eines wissenschaftlichen Textes ist durch diese Beschränkung in der Regel nicht mehr abbildbar.

[4] Das Anlegen eines Google-Kontos ist kostenlos und ermöglicht den Zugriff auf eine Reihe kostenloser wie kostenpflichtiger Dienste von Google. Google Text ist eine kostenloser Dienst.

Das frei verfügbare webbasierte *Writeboard* (writeboard.com) sowie das kommerzielle *ZohoWriter* (writer.zoho.com/home?serviceurl=%2Findex.do) sind ähnliche KS-Systeme.

Mit *Office Web App* (office.microsoft.com/de-de/web-apps) bietet Microsoft webbasierte Varianten der Anwendungen Word, Excel und PowerPoint an. Allerdings ist deren Funktionsumfang gegenüber den Desktop-Versionen stark reduziert. Insbesondere die fehlende Möglichkeit unter Office Web Apps Word-Dokumente gleichzeitig von verschiedenen Autoren editieren zu können stellt einen nicht unerheblichen Nachteil gegenüber Google Text dar, weshalb auf eine ausführlichere Darstellung dieser web-basierten Textverarbeitung verzichtet werden soll[5].

3.2.5 Wiki-Systeme

Ein *Wiki* ist eine Web-Seite, die von jedem Betrachter zu jeder Zeit beliebig verändert werden kann. Mit einem Wiki können Texte kollaborativ in Artikelform erstellt und miteinander verknüpft werden. Typische Funktionen eines Wiki sind:

- Erzeugen eines neuen Artikels zu einem beliebigen Thema
- Ändern und Löschen von Artikeln anderer Autoren
- Erzeugen von Verweisen zwischen Artikeln
- Zusammenfassen von inhaltlich nahestehenden Artikeln zu Kategorien
- Nachvollziehen von Änderungen an einem Artikel
- Erzeugen von Statistiken für Artikel (z.B. Popularität)

Das erste wirkliche Wiki[6] wurde 1995 vom US-amerikanischen Software-Entwickler Ward Cunningham unter dem Namen WikiWikiWeb (c2.com/cgi/wiki) der Öffentlichkeit zur Verfügung gestellt. Seit dem wurden zahlreiche weitere Wikis im Web veröffentlicht. Das bekannteste Wiki ist die Online- Enzyklopädie Wikipedia (www.wikipedia.org). Technologische Grundlage für Wikipedia ist das Wiki-System *MediaWiki* (www.mediawiki.org). *MediaWiki* besitzt folgende zusätzliche kollaborative Fähigkeiten:

- ***Zugriffsrechte:*** In *MediaWiki* können Benutzerrechte (z.B. Erzeugen, Bearbeiten, Löschen und Lesen von Artikeln) für einzelne Benutzer oder Benutzergruppen zugewiesen werden

[5] Es sei angemerkt, dass Office Web Apps noch eine relative junge Sammlung web-basierter Anwendungen ist und eine Fortentwicklung der kollaborativen Fähigkeiten in der Zukunft sehr wahrscheinlich ist.

[6] Wiki ist das hawaiische Wort für „schnell“.

- ***Konflikterkennung:*** In *MediaWiki* kann ein und derselbe Artikel von unterschiedlichen Autoren editiert werden. Geschieht dies zeitgleich, gilt die Regel, dass der Autor, der zuerst seine Änderungen am Artikel speichert, Erfolg hat, während der zweite Autor gezwungen wird, seine Änderungen manuell einzufügen.

3.2.6 Forschung

Es existieren zahlreiche Veröffentlichungen zum Thema *kollaboratives Schreiben*. Neben dem besseren Verständnis und dem Erstellen von Anforderungsprofilen, werden vor allem die technologischen Herausforderungen näher untersucht. Beispiele dafür sind Arbeiten zur Konfliktvermeidung, -erkennung bzw. -auflösung (z.B. Operational Transformation[7]), Arbeiten zur besseren Integration von KS in bestehende Systeme, sowie die konkrete Entwicklung von KS-Systemen. Im Folgenden sollen einige dieser KS-Systeme vorgestellt werden.

In [McAlpine et al. 94] wird eine Architektur für ein kollaboratives Textverarbeitung vorgestellt. Die kollaborative Textverarbeitung *Collaborwriter* soll die Funktionalität (z.B. Formatierungen, Rechtschreibprüfung) und die Textstruktur (Fließtext, hierarchisch gegliedert) gängiger Textverarbeitungssysteme beinhalten. Das Dokumentenformat soll SGML-basiert sein und in unterschiedliche Publikationsformate überführt werden können. *Collaborwriter* soll Konflikte erkennen und auflösen. Dokumente sollen versioniert und Änderungen nachverfolgt werden können. Die Software soll einfach zu bedienen und anpassbar sein. *Collaborwriter* ist meines Wissens nicht aus der Design-Phase herausgekommen.

In [Meier 03] wird im Rahmen einer Diplomarbeit die kollaborative Textverarbeitung *ShaDoW* (Shared Document Writer) vorgestellt. Sie ist in Form eines Prototyps implementiert. *ShaDoW* ist ein dezentrales System, d.h. alle Instanzen müssen sich kennen und tauschen ihre Daten stets direkt untereinander aus. Texte können parallel editiert werden. Das Konflikt-Management wird durch eine adaptive Partitionierung des Textes in Kombination mit optimistischem Sperren umgesetzt. Konkret bedeutet dies, dass nur der gerade zu editierende Teiltext gesperrt wird. Sollten zwei Autoren dennoch an genau derselben Stelle Änderungen am Text vornehmen, wird dieser Konflikt beim nächsten Synchronisieren des Teiltextes erkannt.

CoWord ist eine kollaborative Erweiterung für Microsoft Word in den Versionen 2000, XP (2002) und 2003. Grundlage für die Entwicklung sind die in [Xia et al. 04] und [Shen et al. 06] dokumentierten Forschungsergebnisse. Mit Hilfe von *CoWord*

[7] Operational Transformation (OT) ist eine Technologie zur Konfliktvermeidung bei zeitgleicher Bearbeitung eines Textes durch mehrere Autoren. Ein Beispiel soll die Problematik verdeutlichen: Gegeben ist der Text „abc“. Autor A fügt ein „x“ an Position 2 ein, während gleichzeitig Autor B den Buchstaben an Position 3 löschen möchte. Ohne OT würde aus „abc“ ein „axbc“ und anschließend ein „axc“ werden, oder umgekehrt aus „abc“ ein „ab“ und anschließend ein „axb“ werden. Mit OT würde der resultierende Text immer „axc“ lauten (für eine ausführliche Diskussion siehe Kapitel 4.4.3.1).

können mehrere Nutzer in Echtzeit gemeinsam an einem Text arbeiten. Dazu wird Word um folgende Funktionalitäten erweitert:

- *Angepasster Befehl „Speichern":* Der Speichern-Befehl erlaubt das Speichern des aktuellen Dokuments in einer zentralen Dokumentenablage im Netzwerk.

- *Angepasster Befehl „Rückgängig":* Der Rückgängig-Befehl erlaubt wahlweise die Rücknahme lokaler als auch globaler Befehle. Lokale Befehle sind ausschließlich die Befehle des lokalen Nutzers. Globale Befehle sind die Befehle aller kooperierenden Nutzer

- *Action Modes:* Der Nutzer kann zwischen einem Single-Actor- und Multi-Actor-Modus wechseln. Im Multi-Actor-Modus können alle Nutzer gleichzeitig editieren, im Single-Actor-Modus kann ein ausgewiesener Nutzer (Action Controller) den Text editieren.

- *View Modes:* Der Nutzer kann zwischen einem Single-Viewer-Modus und einem Multi-Viewer-Modus wechseln. Im Multi-Viewer-Modus kann jeder Nutzer jeden Teil des Textes unabhängig von anderen Nutzern betrachten. Im Single-Viewer-Modus müssen alle Nutzer den gleichen von einer ausgewiesenen Nutzer (View Controller) vorgegeben Textausschnitt betrachten.

- *Private Mode:* Im Private Mode werden lokale Änderungen am Text nicht an andere Nutzer propagiert, allerdings werden weiterhin die Operationen der anderen Benutzer lokal ausgeführt.

- *Farbliche Hervorhebungen:* Durch farbliche Hervorhebungen werden die Änderungen unterschiedlicher Nutzer mit unterschiedlichen Farben markiert.

Die technologische Grundlage für CoWord ist eine Software-Bibliothek namens Generic Collaboration Engine (GCE).

3.3 Anforderungen an KS-Software

3.3.1 Anforderungen aus der Forschung

In [Noël et al. 04] wurden insgesamt 41 Personen unter Zuhilfenahme eines Fragebogens u.a. nach Wünschen bezüglich eines idealen KS-Werkzeugs befragt. Dabei ergab sich die folgende Reihenfolge beginnend mit den am häufigsten genannten Wünschen hin zu den am wenigsten genannten Wünschen:

- Synchronisierter Zugriff auf Dokumente

- Einfache Möglichkeit zu kommunizieren

- Einfache Möglichkeit Texte mit Kommentaren zu versehen

- Einfache Nachvollziehbarkeit von Textänderungen
- Einfaches Erstellen eines Zeitplans
- Einfache Möglichkeit Ideen auszudrücken
- Anlegen von Notizen
- Benachrichtigen bei Veränderungen am Text
- Projektplanung

In einer anderen Studie [Tammaro et al. 97] wurden in den 90er Jahren 36 an geografisch verteilten Orten beheimatete Mitarbeiter der MITRE Cooperation (www.mitre.org) über einen Zeitraum von drei Monaten bei der Nutzung einer neu eingeführten KS-Software beobachtet und anschließend befragt. Vor Beginn dieser Studie wurden die Anforderungen für ein solches System abgefragt, um anschließend ein passendes System auszuwählen. Dabei wurden die folgenden Anforderungen identifiziert:

- Die Software muss neben Text, auch das Erstellen von Tabellen und Präsentationen unterstützen.
- Texte, Tabellen oder Präsentationen müssen mit Kommentaren versehen werden können.
- Die Software muss unterschiedliche Rollen für die Gruppenmitglieder unterstützen (Wer darf was lesen oder schreiben?)
- Die Software muss in der Lage sein, unterschiedliche Versionen eines Dokuments speichern zu können.
- Dokumente müssen direkt (ohne Vermittlung eines weiteren Gruppenmitglieds) änderbar sein und die Änderungen müssen für alle anderen Gruppenmitglieder nachvollziehbar gemacht werden. Alle Änderungen müssen mit ihrem Autor ausgewiesen werden.
- Beiträge aus unterschiedlichen Herkunftsquellen sollen einfach eingemischt werden können.
- Wird ein neues Dokument zur Verfügung gestellt oder wird ein bestehendes Dokument geändert, sollen alle beteiligten Gruppenmitglieder, die Zugriff auf das Dokument haben, automatisch darüber benachrichtigt werden.
- Für jedes Dokument soll ein Workflow definiert und auch nachträglich verändert werden können. Dokumente sollen anschließend an Hand dieses Workflows automatisch verarbeitet werden. Die Prozesse, die ein Dokument aufgrund des Workflows durchläuft, müssen nachvollziehbar sein.

In [Jones 95] werden folgende Richtlinien für das Design und die Implementierung einer generischen KS-Software vorgeschlagen:

- Hierarchisierung von Strukturen und Aufgaben
- Unterschiedliche Sichtweisen auf ein und dieselbe Information (z.B. mit oder ohne Anmerkungen)
- Einfacher Wechsel zwischen diesen Sichtweisen
- Erzeugen und Bearbeiten von Notizen und ganzen Netzwerken von Notizen
- Unterstützung von Text und Meta-Text (z.B. Anmerkungen)
- Eindeutige Identifizierung von Gruppenmitgliedern
- Kommunikationsmöglichkeiten zwischen Gruppenmitgliedern
- Versionskontrolle
- Aufteilung eines Dokuments in Segmente
- Einmischen extern erstellter Texte
- Nachvollziehbarkeit aller Aktionen (Historie)
- Unterstützung von Einzelautoren wie auch mehreren Autoren

Aktuellere Forschungsergebnisse, insbesondere vor dem Hintergrund der neueren technischen Möglichkeiten wie z.B. schnelles Internet und aktueller Kollaborationswerkzeuge wie *Google Text* und *Microsoft Word*, sind praktisch nicht vorhanden. Eine Tatsache, welche die folgenden Kapitel motiviert, die eine erweiterte Klassifikation von KS-Systemen aus heutiger Sicht geben.

3.3.2 Anforderungen aus Sicht von KS-Rollen

Während im vorherigen Kapitel durch Feldstudien, bei denen Anwender nach ihren Erfahrungen und Wünschen bezüglich eines kollaborativen Textverarbeitungssystems befragt wurden, Anforderungen extrahiert wurden, soll in diesem Abschnitt ein anderer Weg verfolgt werden. Ausgangspunkt diesmal sind die in Kapitel 2.4 beschriebenen KS-Rollen, die Teammitglieder einnehmen können. Die Bedürfnisse[8] eines Teammitglieds können dabei in Konflikt mit den Bedürfnissen anderer Teammitglieder treten. Dieses Konfliktpotential ist ein wesentlicher Aspekt in der Betrachtung kollaborativer Textverarbeitungssysteme, aus denen sich Anforderungen

[8] Diese Sichtweise leitet sich auch aus dem Software-Architekturstandard IEEE-1471 [IEEE 1471-2000] ab, der als Ausgangspunkt einer jeden Architekturbeschreibung die Bedürfnisse (Concerns) der Beteiligten (Stakeholders) in den Mittelpunkt rückt.

ableiten lassen. Den möglichen Rollen Autor, Berater, Editor, Gutachter, Teamleiter und Lektor lassen sich folgende Bedürfnisse zuordnen:

Rolle	Bedürfnisse
Autor	• Zeit- und ortsunabhängiger Zugang zum Gesamttext bzw. zu Teiltexten. • Zeit- und ortsunabhängige Bearbeitung eigener Teiltexte. • Schutz vor Veränderung eigener Teiltexte durch andere Teammitglieder. • Klare und unveränderbare Identifizierung der eigenen Autorenschaft
Berater	• Zeit- und ortsunabhängiger Zugang zum Gesamttext bzw. zu relevanten Teiltexten. • Einfache und effiziente Kommunikationsmöglichkeit mit Teamleiter, relevanten Autoren und anderen Beratern
Editor	• Zeit- und ortsunabhängiger Zugang zum Gesamttext bzw. zu Teiltexten. • Zeit- und ortsunabhängige Bearbeitungsmöglichkeiten des Gesamttextes bzw. von Teiltexten. • Einfache und effiziente Kommunikationsmöglichkeit mit Teamleiter und Autoren
Gutachter	• Zeit- und ortsunabhängiger Zugang zum Gesamttext • Einfache und effiziente Kommunikationsmöglichkeit mit Teamleiter
Teamleiter	• Muss in der Lage sein, ein KS-Projekt zu starten und zu beenden. • Muss in der Lage sein, neue Teammitglieder aufzunehmen und bestehende Teammitglieder zu entfernen. • Muss in der Lage sein, die Rollenzuweisung an Teammitglieder zu verändern. • Zeit- und ortsunabhängiger Zugang zum Gesamttext. • Einfache und effiziente Kommunikationsmöglichkeit mit allen Teammitgliedern

Rolle	Bedürfnisse
Lektor	• Zeit- und ortsunabhängiger Zugang zum Gesamttext bzw. zu Teiltexten. • Zeit- und ortsunabhängige Bearbeitungsmöglichkeiten von Teiltexten. • Einfache und effiziente Kommunikationsmöglichkeit mit Teamleiter und Autoren

Tabelle 3: KS-Rollen und ihre Bedürfnisse

Bei genauerer Betrachtung der KS-Rollen und ihrer Bedürfnisse lässt sich folgendes feststellen:

- Ein Team wissenschaftlicher Autoren zeichnet sich oft dadurch aus, dass jeder Autor ein Spezialist auf seinem Fachgebiet ist und nur er die Teiltexte bzw. Teilthemen editiert, für die er die Sachkompetenz hat und für die er als Autor auch vermerkt wird, um spätere Korrekturen und Ergänzungen durch denselben Autor zu ermöglichen und um die Grundlage für korrektes Zitieren (Nachvollziehbarkeit wissenschaftlicher Arbeiten) zu gewährleisten. Ein Team von Architekten hingegen, das einen Ausschreibungstext für ein Bauvorhaben verfasst, kann den Ausschreibungstext vom Prinzip beliebig unter sich aufteilen, wobei der Autor eines Ausschreibungsteiltextes nicht gesondert vermerkt werden muss (jeder Architekt kann z.B. die Rohbauausschreibung verstehen und somit ändern).

- Ein Teammitglied kann in mehreren Rollen agieren. So kann beispielsweise der Teamleiter auch gleichzeitig Autor sein, der Berater kann gleichzeitig auch Lektor sein.

- Eine Rolle kann durch mehrere Teammitglieder besetzt sein. Die Rolle Autor gehört typischerweise dazu.

- Nicht alle Rollen müssen zwingend besetzt sein. Die Rolle Berater wäre ein typisches Beispiel hierfür.

- Bestimmte Bedürfnisse sind für alle Rollen gültig (z.B. Orts- und zeitunabhängiger Zugriff), manche Bedürfnisse sind nur für einzelne oder einige wenige gültig (z.B. Identifikation der Autorenschaft).

- Einige Rollen haben die gleichen Bedürfnisse (z.B. Editor und Lektor) und können daher zusammengefasst werden:

    - Der Teamleiter bestimmt die Teammitglieder und weist ihnen ihre Rollen zu.

- Der Gesamttext wie auch Teiltexte muss zu zeit- und ortsunabhängig zugreifbar sein.
- Eigene Teiltexte können von Autoren zeit- und ortsunabhängig bearbeitet werden.
- Eigene Teiltexte müssen vor unerlaubter Veränderung von anderen Teammitgliedern geschützt werden.
- Alle Teammitglieder müssen auf einfache und effiziente Weise miteinander kommunizieren können.

Diese Betrachtung ist sicherlich nicht erschöpfend, sie zeigt jedoch sehr gut das Interaktions- und Interessengeflecht zwischen allen Beteiligten.

3.3.3 Ableitung einer Taxonomie

Der bisherige Blick auf mögliche Anforderungen fehlt ohne Zweifel eine gewisse Systematik, handelt es sich mehrheitlich um bloße Aufzählung von möglichen Anforderungen. Im Folgenden soll daher ähnlich der in Kapitel 2.4 vorgestellten Taxonomie für kollaboratives Schreiben eine korrespondierende Taxonomie für das Anforderungsprofil von KS-Systemen entwickelt werden. Ausgangspunkt hierfür sind die Anforderungen aus den Kapiteln 3.3.1 und 3.3.2 sowie die Funktionalität kollaborativer Textverarbeitungssysteme aus Kapitel 3.2.

3.3.3.1 Klassifikation

Betrachtet man all die vorgeschlagenen Anforderungen, so lassen sie sich meiner Meinung nach wie folgt klassifizieren:

1. **Organisation:** Texte müssen entsprechend eines Zugriffsystems allen Akteuren entsprechend ihrer zugedachten Rollen zugänglich gemacht werden, sie müssen bei Bedarf sinnvoll in Teiltexte zerlegt werden können und ihr Workflow sollte definierbar und beinflussbar sein. Zeitliche (z.B. Veröffentlichungstermin) und formale (z.B. maximale Seitenanzahl) Rahmenbedingungen müssen planbar und kontrollierbar sein. Texte lassen sich aufteilen in Inhalt, Layout sowie begleitende Ressourcen (z.B. Grafiken, Daten)

2. **Kommunikation:** Insbesondere bei asynchronen KS-Arbeitsmodi sind Kommunikationswerkzeuge für alle Gruppenmitglieder wichtig. Dazu zählen E-Mail, Chat und Telefon. Aber auch die Möglichkeit Kommentare, Bewertungen, Meinungen, Ideen und Notizen für alle sichtbar ablegen zu können sowie ein Benachrichtigungssystem für Systemereignisse.

3. **Synchronisation:** Verteilt erstellte Texte müssen zusammengeführt werden. Dazu benötigen die Gruppenmitglieder Werkzeuge zur Konfliktvermeidung (Wer darf wann was ändern?), Konflikterkennung (Wer hat wann was geändert?), Konfliktauflösung (Welche Änderung gewinnt?).

4. **Nachvollziehbarkeit:** Die Arbeit eines jeden Gruppenmitglieds muss eindeutig als seine gekennzeichnet werden können. Dazu werden Werkzeuge zur Versionskontrolle, zum Aufzeichnen von Änderungen (Logbuch) und zur eindeutigen Identifikation (Benutzerprofile) benötigt.

5. **Schutz:** Bedingt durch die Tatsache, dass Texte über zum Teil öffentliche Netzwerke ausgetauscht und synchronisiert werden, müssen die Authentizität von Texten sowie deren Schutz vor Einsicht durch eine nicht gewünschte Öffentlichkeit sichergestellt werden.

Diese Klassifizierung enthält wiederum einige Begriffe, die genauer betrachtet werden sollen.

3.3.3.2 Inhalt, Layout und Ressourcen

Textsatzsysteme wie LaTeX oder TUSTEP liegen auch heute noch bei vielen Wissenschaftlern hoch im Kurs, wenn es darum geht, wissenschaftliche (insbesondere naturwissenschaftliche) Texte zu verfassen. Der Grund liegt in der sauberen und standardisierten Druckausgabe, welche durch diese Systeme automatisch erstellt wird. Dies wird durch eine strikte Trennung von Inhalt und Layout erreicht. Der Autor gibt lediglich vor, um welches Strukturelement es sich bei einem Teiltext handelt, die visuelle Umsetzung wird später durch die Software mit Hilfe von vorgegebenen Regeln durchgeführt. Diese strikte Trennung von Text und Layout lässt sich in modernen Textverarbeitungssystemen wie Word 2010 oft nur sehr unzureichend umsetzen. Generell gibt es zwei mögliche Ansätze, um dieses Ziel zu erreichen:

1. Eine Sperrung der vorhandenen Formatierungsoptionen bis zu einem Grad, der lediglich die Zuweisung eines Formates zu einem Text ermöglicht, nicht aber die Bearbeitung des Formates als solches.

2. Ein Verlagerung der Formatierung auf in einen Post-Prozess, der erst nach Fertigstellung des Textes angestoßen wird, um die gewünschte Publikationsform zu erstellen.

Beide Ansätze unterscheiden sich dahingehend, dass der erste Ansatz dem Autor bereits während der Texterstellung das resultierende Layout vor Augen führt (WYSIWYG-Prinzip), während der zweite Ansatz dies vor dem Autor verbirgt. Wichtig ist jedoch in beiden Fällen, dass der Autor keine Möglichkeit bekommt, das Layout zu ändern. Dies entspricht zum einem dem Rollenverständnis zwischen Autor und Layouter, und es dient im erheblichen Maße der Konfliktvermeidung.

Die Betrachtung des Textes, bestehend aus Inhalt und Layout, als alleiniger Kollaborationsgegenstand ist zu kurz gefasst. Insbesondere die elektronische Publikation ermöglicht es Dokumente zu erstellen, die über den sequenziellen Textcharakter der klassischen Papierform hinausgehen: Eine Textkollaboration ist ebenfalls abhängig von externen Ressourcen, die direkt in den Text eingebunden werden oder aber als Informationsquelle für die Texterstellung dienen. Mögliche Ressourcen wären beispielsweise Grafiken, Fotos, Videos, Tondokumente, Textdokumente und

Forschungsdaten. All diese Ressourcen sind genauso Teil der Kollaboration wie der eigentliche Text. Daher lassen sich folgende Anforderungen ableiten:

- Ressourcen müssen unabhängig von Ort und Zeit jedem Autor zur Verfügung stehen.
- Ressourcen müssen eindeutig identifizierbar und recherchierbar sein.
- Ressourcen müssen einer Versionierung unterliegen.
- Der Zugriff auf Ressourcen muss einschränkbar sein.

Die Forderung nach einer Hochverfügbarkeit von Ressourcen ergibt sich aus der gleichen Forderung bei Texten. Da Ressourcen zum Teil durch Verweise direkt in den Text eingebunden werden können, müssen sie in gleicher Weise zur Verfügung stehen wie der Text. Je größer die Menge der verwendeten Ressourcen ist, desto wichtiger sind eine eindeutige Identifizierung der Ressource sowie deren Recherchierbarkeit. Identifizierbarkeit bedeutet, dass Name, Zweck und Ersteller hinterlegt sind. Recherchierbarkeit bedeutet, dass eine effektive Suche über alle verfügbaren Ressourcen möglich sein sollte. Wie Texte sollten auch Ressourcen einer Versionierung unterzogen werden, wenn sie sich über die Zeit ändern (z.B. Anpassung einer Grafik).

3.3.3.3 Verfügbarkeit

Als Einzelautor hat man seinen Text in der Regel auf seinem eigenen Computer gespeichert und damit eine uneingeschränkte Verfügbarkeit des Textes. Sobald man sich vor den Computer sitzt, steht auch der Text zur Verfügung. Im kollaborativen Umfeld ändert sich jedoch der Text auch dann, wenn ein Autor nicht vor seinem Computer sitzt. Dies impliziert die folgenden zwei Forderungen:

- Ein Text muss unabhängig von Ort und Zeit unmittelbar zur Verfügung stehen.
- Eigene Änderungen müssen unabhängig von Ort und Zeit unmittelbar allen anderen Teammitgliedern zur Verfügung gestellt werden können.

Die Charakterisierung „unabhängig von Ort und Zeit“ besagt, dass lediglich der Zugriff auf das Internet nötig sein sollte, um an den gemeinsamen Text zu gelangen. Dies wiederum setzt eine Hochverfügbarkeit eines KS-Systems voraus. Die aktuellen Angebote im Bereich Cloud-Services (z.B. Microsoft Azur oder Amazon S3) erlauben schon heute, eine einfache Implementierung von Hochverfügbarkeit. Diese Forderung ist beispielsweise vor dem Hintergrund stark variierender Systemvoraussetzungen bei Wissenschaftlern zu sehen. Vor allem im nicht-naturwissenschaftlichen Bereich ist das Wissen über das Einrichten und Administrieren zentraler Server-Infrastrukturen nur unzureichend vorhanden und oft auch gar nicht bezahlbar. Es bleibt also nur das Ausweichen auf bereits etablierte Dienstleistungen im Internet.

3.3.3.4 Konflikte

Schreiben mehrere Autoren an einem Text, so entsteht das Problem der Textsynchronisation. Theoretisch kann ein Autor den Text eines anderen Autors verändern oder gar löschen. Da dies in den meisten Fällen unerwünscht bzw. unter allen Umständen zu vermeiden ist, ist das Interesse eines Autors sehr hoch, dass sein Textbeitrag nur von ihm selbst verändert oder gelöscht werden darf. Es können prinzipiell zwei Arten von Konflikten auftreten:

1. **Synchronisationskonflikte:** Mehrere Autoren bearbeiten gleichzeitig denselben Teiltext.

2. **Zugriffskonflikte:** Ein Beteiligter, z.B. ein anderer Autor, ein Editor, ein Lektor, möchte den Text eines Autors nachträglich ändern.

Die Gründe dafür können sein:

- Durch eine unscharfe Abgrenzung des Inhalts unter den beteiligten Autoren, muss ein Autor den Text eines anderen Autors bearbeiten.

- Ein Lektor muss in der Lage sein, einen Text bezüglich Rechtschreib- und Grammatikfehlern korrigieren zu können.

- Ein Projektleiter muss in der Lage sein, Textbeiträge von Autoren abzulehnen oder zu löschen, wenn Sie bestimmten Qualitätskriterien nicht genügen oder der Autor sich aus dem Projekt vorzeitig zurückgezogen hat.

Konflikte bedürfen eines Konflikt-Managements, das sich wie folgt charakterisieren lässt:

- **Konfliktvermeidung** – Hierbei geht darum, das Konfliktpotential von vorneherein auf ein Minimum zu reduzieren. Dies kann beispielsweise dadurch geschehen, dass die Anzahl der Autoren minimiert wird oder aber der Schreibzugriff möglichst exklusiv vergeben wird.

- **Konflikterkennung** – Ist dann doch ein Konflikt entstanden, so muss er zunächst erkannt werden, d.h. die Autoren oder der Projektleiter müssen ein Feedback bekommen, dass sich der Text momentan in einem inkonsistenten Zustand befindet und akuter Handlungsbedarf besteht.

- **Konfliktauflösung** – Ein erkannter Konflikt muss aufgelöst werden, d.h. der inkonsistente Zustand des Textes muss wieder in einen konsistenten Zustand überführt werden.

Für eine ausführliche Behandlung des Themas Konflikt-Management auf technischer Ebene sei auf das Kapitel 4.4 verwiesen.

3.3.3.5 Authentizität

Der Befürchtung mancher Autoren, dass sie innerhalb einer kollaborativen Texterstellung der gewollten oder ungewollten Manipulation der eigenen Texte schutzlos ausgeliefert sind, lässt sich durch den Einsatz elektronischer Signaturen entgegenwirken. Mit Hilfe einer elektronischen Signierung lassen sich die Integrität und die Authentizität eines digitalen Textes bzw. Teiltextes sicherstellen. Ähnlich einer Unterschrift bei herkömmlichen Texten in Papierform wird hierbei der bestehende Text um eine elektronische Signatur erweitert. Das ganze Verfahren basiert auf einer asynchronen Verschlüsslungstechnik und funktioniert wie folgt: Der Autor eines Textes ist im Besitz eines kryptographischen Schlüsselpaars, bestehend aus einem privaten Schlüssel (Private Key) und einem öffentlichen Schlüssel (Public Key). Der private Schlüssel ist geheim und nur dem Autor bekannt, der öffentliche Schlüssel dagegen wird der Öffentlichkeit zur Verfügung gestellt. Zwischen beiden Schlüsselwerten besteht eine mathematische Abhängigkeit, die zur Folge hat, dass Informationen, die mit einem privaten Schlüssel verschlüsselt werden, nur mit dem dazu passenden öffentlichen Schlüssel wieder entschlüsselt werden können.

Möchte der Autor seinen Text elektronisch signieren, so lässt er sich zunächst mit Hilfe einer Hash-Funktion eine Prüfsumme für diesen Text errechnen. Diese Prüfsumme dient als digitaler Fingerabdruck des Textes. In einem zweiten Schritt wird diese Prüfsumme mit Hilfe des privaten Schlüssels verschlüsselt und an den bestehenden Text angehängt. Der Text besitzt jetzt eine elektronische Signatur. Der Leser des Textes hat nun die Möglichkeit nachzuprüfen, ob:

a) die Signatur auch wirklich vom angegebenen Autor stammt (Authentizitätsprüfung)

b) der Inhalt des Textes verändert wurde (Integritätsprüfung)

Dazu entschlüsselt er zunächst die angehängte Prüfsumme mit Hilfe des öffentlichen Schlüssels des Autors. Gelingt dies, so weiß der Leser, dass die Prüfsumme mit dem privaten Schlüssel des Autors verschlüsselt wurde, die Signatur also tatsächlich vom Autor stammt. Anschließend lässt der Leser erneut die Prüfsumme des Dokumentes errechnen und vergleicht sie mit der soeben entschlüsselten Prüfsumme. Ist der Vergleich positiv, so weiß der Leser, dass der Inhalt nach der Signatur durch den Autor nicht mehr verändert wurde.

3.4 Analyse: Microsoft Word und Google Text

Dieser Abschnitt analysiert die in Kapitel 3.2.4 vorgestellten Textverarbeitungssysteme *Microsoft Word 2010* und *Google Text* anhand der KS-Anforderungen. Beide stellen unter den in dieser Arbeit erwähnten KS-Systemen die einzigen öffentlich verfügbaren Lösungen dar. Sie werden in den folgenden Abschnitten jeweils parallel betrachtet.

Viele wissenschaftliche Publikationen werden heute mit Hilfe der Textverarbeitung Microsoft Word erstellt. Dies liegt zum einen daran, dass Word wohl die statistisch

gesehen am häufigsten eingesetzte Textverarbeitung ist und somit deren Bedienung den meisten Autoren geläufig ist. Zum anderen ist der Leistungsumfang von Word ausreichend, um auch komplexe Textdokumente verwalten zu können. Der eigentliche Grund, warum Word Gegenstand dieses Kapitel ist, liegt jedoch in den kollaborativen Fähigkeiten der aktuellen Version 2010. Diese ist ein beachtlicher Schritt, da es als das wohl prominenteste Textverarbeitungsprogramm den Gegensatz zwischen komplexer, für Einzelautoren ausgelegter Textverarbeitung auf der einen Seite und reduzierter, für paralleles Schreiben ausgelegter Textverarbeitung auf der anderen Seite zu schließen beginnt. Eine Feststellung, die insofern Beachtung verdient, da alle bisher untersuchten Systeme entweder zu wenig kollaborativ sind (z.B. Word 2007) oder aber zu wenig Textverarbeitung bieten (z.B. Google Text). In den folgenden Abschnitten soll die kollaborativen Fähigkeiten dieser Programme anhand der zuvor gemachten Überlegungen untersucht werden.

Microsoft Word 2010 erlaubt es zwei oder mehreren Benutzern auf einem gemeinsamen Textdokument zu arbeiten. Der Benutzer hat dabei die Wahl mit einer Web-Browser-Version von Word oder einer unter dem Betriebssystem Windows installierten Desktop-Version von Word zu arbeiten. Da die kollaborativen Fähigkeiten der Browser-Version nicht so ausgeprägt sind wie jene der Desktop-Version (beispielsweise kann ein Dokument zu einem Zeitpunkt nur exklusiv von einem Benutzer bearbeitet werden), soll sich der Fokus auf die in diesem Zusammenhang interessantere Desktop-Version richten.

Google Text ist eine web-basierte Textverarbeitung. Während Microsoft Word auch für umfangreiche wissenschaftliche Texte eingesetzt wird, erlaubt der weitaus geringere Funktionsumfang von Google Text, wie z.B. fehlende Vorlagen, praktisch keine Unterstützung von Quellenverzeichnissen, es gar nicht erst, Google Text für umfangreichere Veröffentlichungen zu verwenden. Da es sich für weniger komplexe Texte durchaus eignet und im Gegensatz zu Microsoft Word einen originär Webbrowser-basierten Ansatz darstellt, soll Google Text im Folgenden dennoch parallel betrachtet werden.

3.4.1 Verfügbarkeit von Texten

Bei Microsoft Word ist durch die Nutzung von *SkyDrive*[9] eine Hochverfügbarkeit des Textes gewährleistet. Selbst externe Ressourcen können unter SkyDrive abgelegt werden. Allerdings setzt dies eine ständige Verfügbarkeit einer funktionierenden Internet-Verbindung voraus. Zumindest in Deutschland sollte dies in den überwiegenden Fällen kein Problem darstellen. Selbst das mobile Arbeiten in der Bahn ist heutzutage durch Nutzung von UMTS/GPRS möglich.

Word 2010 besitzt jedoch eine Eigenschaft, welche die Verfügbarkeit des Textes an einem anderen Punkt im Netzwerk einschränken kann. Da Word 2010 lediglich unter Windows lauffähig ist, führt der Einsatz von anderen Betriebssystemen zwangsläufig zu einer Einschränkung der Verfügbarkeit. Ein Fakt, der gerade im wissenschaftli-

[9] Siehe Abschnitt *3.2.1 Klassische Textverarbeitung*

chen Bereich, nicht zu unterschätzen ist, da dem Einsatz von Microsoft-Produkten generell eine gewisse Ablehnung (unter anderem wegen deren kommerziellen Natur) entgegengebracht wird. Eine mögliche Abmilderung der Problematik könnte im Einsatz virtueller Maschinen bestehen, die das Betreiben eines Windows-Betriebssystems unter Unix, Linux, MacOS usw. erlauben und damit auch den Einsatz von Word 2010.

Google Text setzt als Web-Anwendung – analog zu Microsoft Word – eine ständig funktionierende Online-Internet-Verbindung voraus. Der Autor benötigt nur einen Webbrowser für seine Arbeit, der praktisch auf allen Plattformen verfügbar ist.

3.4.2 Rechtesystem und Schutz vor Veränderung

Word 2010 erlaubt das explizite Sperren von Textabschnitten. Damit lässt sich eine pessimistische Konfliktvermeidung auf Teiltextebene realisieren. Allerdings ist die Granularität der möglichen Sperrungen nicht ausreichend für die wissenschaftliche Texterstellung. So wäre beispielweise das explizite Sperren von Zitaten, Formeln oder Tabellen wünschenswert.

Google Text erlaubt kein Sperren von Textabschnitten. Es entspricht in seiner Metapher den Wiki-Systemen, die allen Autoren Änderungen erlauben, wobei jeder die Änderungshistorie betrachten und ggf. Änderungen rückgängig machten kann.

Bei beiden fehlt ein Rechtesystem, das ein differenziertes Sperren und Freigeben von Inhalten erst überhaupt ermöglichen würde. So ist bei beiden auch eine automatische Benachrichtigung eines Autors bei Änderungen in Dokumentbereichen nicht vorgesehen.

3.4.3 Synchronisation

Word 2010 unterstützt beide Synchronisationsszenarien, sowohl die Online-Synchronisation als auch die Offline-Synchronisation. Insbesondere die Offline-Synchronisation ist für die wissenschaftliche Texterstellung nach wie vor sehr wichtig. Neben der Tatsache, dass eine ständige Internetverbindung nicht immer möglich ist, bietet nur sie ein wirklich flüssiges Arbeiten mit großen Texten. Es ist leicht zu beobachten, dass die Online-Synchronisation mit zunehmender Größe der Texte an Performanz verliert und somit den Arbeitsfluss erheblich stören kann. Zudem möchte ein Autor nicht ständig seinen Online-Status Preis geben, um keine Rückschlüsse auf seinen Arbeitsrhythmus zuzulassen.

Das Konflikt-Management in Word 2010 ist durchaus als vorbildlich anzusehen. Im Online-Modus, in dem Textänderungen unmittelbar mit dem Server abgeglichen werden, ist jener Teil, der von einem Autor aktuell editiert wird, automatisch für alle anderen Autoren gesperrt. Änderungen, die im Offline Modus aufgelaufen sind, werden beim Synchronisieren mit dem Server automatisch auf Konflikte hin untersucht. Erst nach Auflösung aller Konflikte kann der Autor weiterarbeiten.

Google Text enthält keine Konfliktauflösung. Jeder Autor kann im Gesamtdokument alles jederzeit verändern.

3.4.4 Kommunikationsmöglichkeiten

Als explizites Kommunikationsinstrument bietet Word 2010 das Kommentieren von Teiltexten an. Darüber hinaus lassen sich externe Kommunikationswerkzeuge wie E-Mail, Chat und Internettelefonie mit Word 2010 verknüpfen. Dies setzt allerdings das Vorhandensein solcher Dienste auf dem eigenen Computer voraus. Diese Philosophie des Einbindens externer Dienste ist durchaus kritisch zu sehen. Möchte man sich beispielsweise per Chat austauschen, so müssen alle Teammitglieder die gleiche Chat-Software installiert haben. Und diese muss auch noch kompatibel zu Word 2010 sein (Skype ist es beispielsweise aktuell nicht). Da die Vielfalt heutiger Kommunikationswerkzeuge mittlerweile unüberschaubar geworden ist, kann nicht davon ausgegangen werden, dass zwei Teammitglieder stets die gleichen Werkzeuge benutzen. Die Installation von zusätzlicher Software ist keine zufriedenstellende Lösung. Lediglich E-Mail ist durch seine Standardisierung universell einsetzbar. Es wäre also wünschenswert, wenn Word 2010 bereits Kommunikationsmöglichkeiten von Hause mitbringt. Dies würde neben der sofortigen Verfügbarkeit noch eine Reihe weiterer Vorteile mit sich bringen:

- Es ist keine separate Installation von Software notwendig.
- Die Kommunikation wird automatisch auf den Kreis der Teammitglieder eingeschränkt.
- Nachrichten können um textrelevante Metadaten angereichert werden. So könnten beispielsweise direkte Verweise auf eine bestimmte Textstelle genutzt werden. Oder Formulierungen bzw. vorgeschlagene Korrekturen könnten per Mausklick direkt in den Text übernommen werden.

Google Text bietet dem Autor die Möglichkeit an einer beliebigen Stelle im Text eine Notiz einzufügen. Weitere integrierte Funktionen für die Kommentierung der Notiz durch andere Autoren fehlen in Google Text. Allerdings erlaubt ein Google-Konto die kostenlose Nutzung von E-Mail und Chat

3.4.5 Projektplanung und Zeitpläne

Weder Microsoft Word noch Google Text bieten Unterstützung für Projektplanung und Zeitpläne. Bei Microsoft fällt der Bereich Zeit- bzw. Projektplanung Microsoft Project zu, das wie Word Teil der Microsoft Office Programmsammlung ist. Allerdings gibt es in Microsoft Project keine Integration mit der Textverarbeitung in dem Sinne, dass sich eine direkte Verbindung von Abschnitten im Dokument mit Meilensteinen der Projektsteuerung herstellen ließe. Bei Google Text könnte allenfalls auf den Google-Kalender-Dienst verwiesen werden, der jedem Google-Konto kostenlos zur Verfügung steht.

3.5 Schlussbetrachtung

Viele Anforderungen finden sich schon in Veröffentlichungen zum Thema KS aus den 90er Jahren. Umso erstaunlicher ist es, dass im wissenschaftlichen – wie auch im kommerziellen – Umfeld bis heute kaum geeignete KS-Systeme zu finden sind. Das populärste KS-Werkzeug scheint immer noch E-Mail zu sein, gefolgt von Wiki-Systemen. Der Grund liegt möglicherweise in dem hohen Entwicklungsaufwand für ein solches System.

Für wissenschaftliche Arbeiten wird sehr gerne auf Satzsysteme wie LaTeX und TUSTEP zurückgegriffen, die jedoch keine kollaborativen Fähigkeiten besitzen.

LaTeX ist ein Softwarepaket, das die Benutzung des Textsatzprogramms TeX mit Hilfe von Makros vereinfacht. TeX wurde Ende der 70er Jahre von Donald E. Knuth, Professor an der Stanford-University, entwickelt. Der amerikanische Informatiker Leslie Lamport entwickelte Anfang der 1980er darauf aufbauend eine umfangreiche Sammlung von TeX-Makro-Sammlung, die er in Anlehnung an die beiden Anfangsbuchstaben seines Nachnamens LaTeX nannte. LaTeX interpretiert diese Formatanweisungen (Makros) und erstellt ein druckfertiges Ergebnis, das wahlweise ausgedruckt oder in andere Formate wie Postscript oder PDF konvertiert werden kann. Gerade im mathematisch-naturwissenschaftlichen Bereich hat sich LaTeX als Standard für das Erstellen von Texten etabliert.

Das „Tübinger System von Textverarbeitungs-Programmen“ TUSTEP wird seit den 70er Jahren am Zentrum für Datenverarbeitung der Universität Tübingen mit dem Ziel entwickelt, Autoren ein Software-Werkzeug für die Erstellung bzw. Bearbeitung von wissenschaftlichen Texten zur Verfügung zu stellen. TUSTEP ähnelt LaTeX, da es auch einen mit Makros formatierten Text in ein Ausgabeformat überführt. Allerdings hat TUSTEP einen umfassenderen Ansatz und enthält noch viel mehr Funkionen für die Arbeit mit wissenschaftlichen Texten: z.B. Vergleichen von verschiedenen Textfassungen, Korrigieren anhand vorbereiteter bzw. automatisch erstellter Korrekturanweisungen, Zerlegen von Texten in (frei definierbare) Elemente (z. B. Wortformen), Sortieren von Textelementen oder von längeren Texteinheiten nach beliebigen Alphabeten und einer Vielzahl anderer Kriterien, Register erstellen durch Zusammenfassen sortierter Textelemente etc.

Die ungebrochene Popularität beider Systeme passt eigentlich nicht zur Tatsache, dass heute wissenschaftlichen Veröffentlichungen nur noch sehr selten von einem einzelnen Autor erstellt werden.

Web-basierte Textverarbeitungssysteme (z.B. Google Text) und Wiki-Systeme (z.B. MediaWiki) sind interessante Alternativen zu klassischen Textverarbeitungssystemen, da sie für einen kollaborativen Umgang mit Texten konzipiert sind, allerdings lassen mit diesen Systemen nicht wirklich komplexe wissenschaftliche Texte erstellen. So ist es u.a. in keinem dieser Systeme möglich, Formatvorlagen für eine standardisierte Auszeichnung von Texten zu definieren. Diese sind jedoch zwingend notwendig, um einen Text in eine andere Publikationsform überführen zu können.

Auch das Erstellen von Fußnoten, Inhaltsverzeichnissen, Abbildungsverzeichnissen, Literaturverzeichnissen, Indizes etc. ist nicht oder nur sehr eingeschränkt möglich.

Klassische Textverarbeitungssysteme basieren in der Regel auf dem Grundsatz, dass zu einem beliebigen Zeitpunkt nur ein Nutzer ein bestimmtes Textdokument bearbeiten kann. Es bleiben demnach nur zwei Möglichkeiten eine kollaborative Texterstellung zu unterstützen.

- Der Text wird disjunkte Teiltexte aufgeteilt, jeder Teiltext wird in einer eigenen Datei gespeichert und jedem dieser Teiltexte wird ein ausgewiesener Autor zugwiesen, der für die Erstellung des Inhalts verantwortlich ist. Fast alle Textverarbeitungssysteme unterstützen diese Art der Gliederung.

- Der Text wird als Ganzes von Autor zu Autor weitergereicht. Zu einem Zeitpunkt darf nur ein ausgewiesener Autor am Text schreiben. Diese Art der Aufteilung ist natürlich immer möglich, allerdings können manche Textverarbeitungssysteme dies durch Änderungsprotokollierung und Kommentierungsmöglichkeiten noch weiter unterstützen.

Die erste Strategie erlaubt eine parallele Texterstellung, die zweite Strategie eine sequentielle Texterstellung. Allerdings sind Umsetzungen beider Strategien kritisch zu betrachten:

- Es fehlt eine zentrale Verteilungskoordinierung des Textes bzw. der Teiltexte. Es ist beispielsweise umständlich, parallel zur eigenen Texterstellung die aktuellen Texte der anderen Autoren zu lesen, da diese explizit gefragt werden müssen.

- Eine parallele Texterstellung mit verteilten Rollen oder gar eine reaktive Texterstellung ist nicht möglich.

Keines der betrachteten KS-Systeme kann somit die in Kapitel 3.3 definierten Anforderungen in vollem Umfang umsetzen. In Kapitel 5 werden zwei Ansätze für KS-Systeme vorgestellt, die diese Anforderungen erfüllen.

4 KS-Entwurfsmuster

4.1 Einführung

Bevor in Kapitel 5 zwei neue Entwürfe für eine KS-Software-Architektur vorgestellt werden, welche die in Kapitel 3 diskutierten Anforderungen umsetzen, werden in diesem Kapitel die Bausteine bzw. Entwurfsmuster eines KS-Systemen beschrieben und diskutiert. Diese Entwurfsmuster bilden die Grundlage für die neu eingeführten Architekturen des Kapitels 5.

Entwurfsmuster in der Software-Technik gehen zurück auf die Arbeiten des Architekten Christopher Alexander, der versucht hat, komplexe Architekturstrukturen logisch zu formalisieren und miteinander zu verknüpfen. Diese Idee wurde später vom Autorenquartett Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides aufgegriffen und in ihrem Buch „Design Patterns - Elements of Reusable Object-Oriented Software“ [Gamma et al. 94] auf das Softwareidiom angewandt.

Entwurfsmuster sind allgemeine Lösungsansätze für wiederkehrende Probleme. Für die Lösung ein und desselben Problems kann es unterschiedliche Entwurfsmuster geben, die sich komplementieren oder gegenseitig ausschließen. Manche Entwurfsmuster können nur unter bestimmten Vor- oder Nachbedingungen implementiert werden. Und selbstverständlich schließen diese Entwurfsmuster nicht aus, dass vielleicht noch eine bessere Lösung existiert. Aber eines haben alle Entwurfsmuster gemeinsam: Sie erlauben es, ein weitestgehend abstraktes Vokabular für eine effektive Diskussion von Software-Architekturen zu entwickeln.

„Model-View-Controller“ ist ein populäres Beispiel für ein solches Entwurfsmuster. Es beschreibt die Aufgliederung einer Architektur in ein Modell (model), eine Präsentation (viewer) und eine Steuerung (controller). Das Modell implementiert den Datenzugriff und die Logik der Anwendung, die Präsentation implementiert die grafische Oberfläche und die Steuerung reagiert auf die Eingabe der Präsentation und agiert als Vermittler zwischen Modell und Präsentation. Diese Sichtweise auf eine Software-Anwendung erlaubt eine Trennung von Daten und Logik auf der einen Seite und der grafischen Repräsentation auf der anderen Seite. Dies wiederum ist die Voraussetzung, um Software-Anwendung effizient an wechselnde Repräsentationen anzupassen (z.B. in unterschiedlichen Betriebssystemen).

In den folgenden Kapiteln werden die typischen Entwurfsmuster heutiger KS-Systeme herausgearbeitet. Es existieren scheinbar keine nennenswerte Erörterung zu dieser Thematik, so dass die Rückschlüsse vollends aus der Analyse jener Systeme gezogen wurden. Es werden System-Architekturen, das Text-Management, das Konflikt-Management sowie die Nachvollziehbarkeit von Textänderungen betrachtet.

4.2 System-Architekturen

Eine erste Annäherung an die divergente Vielfalt existierender KS-Systeme ist eine grobe Klassifizierung in dedizierte KS-Systeme, KS-fähige Systeme, und KS-unterstützende Systeme:

1. **Dedizierte KS-Systeme** sind Software-Werkzeuge, die von vorne herein für das kollaborative Erstellen von Texten konzipiert und implementiert wurden. Beispiele hierfür sind Wiki-Systeme sowie Online-Textverarbeitungen wie Google Text.

2. **KS-fähige Systeme** sind Software-Werkzeuge, deren kollaborative Fähigkeiten lediglich einen untergeordneten Teil der Gesamtfunktionalität ausmachen. Oft sind dies Systeme, die nach wie vor auf einen Einzelautor zugeschnitten sind und nachträglich um jene Möglichkeiten erweitert wurden. Beispiele hierfür sind klassische Textverarbeitungssysteme wie Microsoft Word oder Apple iWorks.

3. **KS-unterstützende Systeme** wiederum sind Software-Technologien, die lediglich einen ganz bestimmten Teilaspekt innerhalb einer KS-Gesamtarchitektur abdecken. Beispiele hierfür sind Dokumenten-Management-Systeme oder begleitende Kommunikationstechnologien wie E-Mail.

Die folgenden Entwurfsmuster konzentrieren sich auf KS-Systeme der ersten beiden Klassifizierungen. KS-unterstützende Systeme isoliert zu betrachten macht meines Erachtens wenig Sinn, da sich ihre Bedeutungen erst im Kontext eines KS-Gesamtsystems entfalten. Aus dieser so eingegrenzten Menge von KS-Systemen lassen sich die folgenden vier grundlegenden Architekturmuster extrahieren: Lose gekoppelte Architekturen, Peer-to-Peer-Architekturen, Client-Server-Architekturen und Web-basierte Architekturen:

1. **Lose gekoppelte Architektur** - Hierbei werden voneinander unabhängige Systeme durch einen Austausch von Daten miteinander verbunden. Dieser Datenaustausch kann direkt (z.B. durch eine Import-/Export-Schnittstelle) oder indirekt (z.B. durch Nutzung eines Dokumenten-Management-Systems) geschehen.

2. **Peer-to-Peer-Architektur** - Hierbei schließen sich mehrere gleichberechtigte Systeminstanzen zu einem Netzwerk zusammen, um direkt miteinander zu kommunizieren. Jede Systeminstanz stellt allen anderen Instanzen genau jene Dienste zur Verfügung, die es selbst auch nutzt.

3. **Client-Server-Architektur** - Hierbei kommt es zu einer Aufgabenverteilung innerhalb des Netzwerks. Eine ausgewiesene Systeminstanz (der Server) stellt allen anderen Instanzen (den Clients) alle benötigten Dienste zur Verfügung. Die Clients kommunizieren ausschließlich mit dem Server niemals direkt untereinander im Netzwerk.

4. **Web-basierte Architektur** - Eine Web-basierte Architektur ist eine spezielle Form einer Client-Server-Architektur, die sich der bereits vorhandenen Web-Infrastruktur (Internet-Browser und Web-Server) bedient.

4.3 Text-Management

4.3.1 Die vier Phasen des Text-Managements

Eine wesentliche Aufgabe eines KS-Systems ist die Verwaltung des Textes. Dabei lassen sich vier Phasen ausmachen, die nicht notwendigerweise sequentiell auf einander folgen müssen:

1. Textstrukturierung
2. Textverteilung
3. Textaktivitäten
4. Textsynchronisation

Am Beginn eines jeden KS-Projektes steht immer die Frage, wie soll der resultierende Text am Ende aussehen? Die Autoren müssen sich über inhaltliche Fragen (z.B. Welches Thema?), formale Fragen (z.B. Soll es ein Buch, Artikel oder eine Online-Publikation werden?) und Stilfragen (z.B. wissenschaftlicher vs. informeller Text) abstimmen. Die Beantwortung dieser Fragen führt zu einem ersten Entwurf für eine Strukturierung des Textes. Dies kann eine konkrete Gliederung oder auch nur eine lose Sammlung der zu behandelnden Themen sein. Auf alle Fälle aber ist die Struktur eines Textes stets der Ausgangspunkt für die weiteren Phasen des Text-Managements.

Ist die Textstruktur festgelegt, kann der Text für die parallele oder reaktive Texterstellung unter den Autoren aufgeteilt werden. Hier tun sich mehrere Strategien auf, die je nach Textstruktur besser oder schlechter geeignet sind. So muss z.B. die Frage beantwortet werden, ob jeder Autor immer mit einer Kopie des Gesamttextes arbeitet oder nur mit einem ihm zugewiesenen Teiltext.

Der Textaufteilung folgen die Textaktivitäten, z.B. Ideen sammeln, schreiben, redigieren (siehe KS-Aktivitäten in Kapitel 2.4.3). Durch die Aktivitäten entstehen Änderungen am Gesamttext an verschiedenen Orten durch verschiedene Autoren. Diese Änderungen müssen zu einem späteren Zeitpunkt untereinander synchronisiert werden, so dass jeder Autor die Änderungen seiner Kollegen mitbekommt.

Dieses 4-Phasen-Modell lässt sich bei allen untersuchten KS-Werkzeugen beobachten. Selbstverständlich ist dies weder ein streng sequentielles Modell noch ein Modell mit zeitlich singulären Phasen. Die Textstrukturierung ist einem fortlaufenden Prozess unterzogen. In den seltensten Fällen steht beispielwiese eine Gliederung von vorne herein fest und wird später nicht mehr geändert. Auch die Textaufteilung und Textsynchronisierung meist in kurzen Abständen und fortlaufend wiederholt,

um alle Autoren an der Entstehung des Gesamttextes teilhaben zu lassen. Je umfangreicher der Text ist, desto höher ist auch die Wahrscheinlichkeit, dass Autoren ihre Rollen in der gemeinsamen Texterstellung ändern werden. Neue Autoren können zu einem späteren Zeitpunkt dem KS-Projekt beitreten, bestehende Autoren können das KS-Projekt vorzeitig verlassen, Kompetenzen können sich unter den Autoren verschieben usw. All dies kann wiederum zu neuen Textstrukturen und Textaufteilungen führen.

4.3.2 Textstrukturierung

Nicht jeder Text ist als Gegenstand eines KS-Prozesses geeignet. Ein wesentliches Kriterium dafür ist die Struktur des Textes, die erheblichen Einfluss darauf haben kann, ob und wie ein Text auf mehrere Autoren aufgeteilt werden kann. Es muss zunächst zwischen schwach strukturierten und stark strukturierten Texten unterschieden werden. Schwach strukturierte Texte sind sequentiell zu lesende Texte, die außer Wörtern, Sätzen und vielleicht noch Absätzen keine weiteren Strukturmerkmale aufweisen. Als Beispiel hierfür sei das Manuskript der Rede des Bundespräsidenten Richard von Weizsäcker am 8. Mai 1985 im Bundestag anlässlich des 40. Jahrestages der Befreiung vom Nationalsozialismus erwähnt. Der Inhalt des Textes hat einen monolithischen Charakter, er ist als Ganzes zu betrachten und kann nur sehr schwer in Teiltexte zerlegt werden. Aus diesem Grund und bedingt durch die Tatsache, dass diese Rede an keiner Stelle in Abschnitte unterteilt wurde, ist eine Multi-Autorenschaft nur sehr schwer vorstellbar. Schwach strukturierte Texte werden in der Regel durch Einzelautoren geschrieben, sie sind nicht sehr umfangreich und bieten kaum Ansatzpunkte für eine erfolgreiche KS-Strategie. Dies verändert sich mit zunehmendem Strukturierungsgrad eines Textes. Als Gegenbeispiel zur Rede Weizsäckers sei das DUDEN-Wörterbuch der deutschen Rechtschreibung erwähnt. In der 21. Auflage werden kurze Informationen (neben der korrekten Schreibweise, Silbentrennung und Aussprache sind dies vor allem Angaben zur Etymologie) zu mehr als 115.000 Stichwörtern gegeben. Ein solcher Text kann als stark strukturiert angesehen werden. Jedes Stichwort und seine zugehörigen Erklärungen bildet einen klar abgegrenzten Teiltext, der zudem als weitestgehend unabhängig von allen anderen Teiltexten in diesem Buch zu betrachten ist. Alle möglichen KS-Strategien lassen sich problemlos auf diesen Text anwenden und in der Tat ist das DUDEN-Wörterbuch auch das Gemeinschaftswerk einer ganzen Reihe von Linguisten.

Zusammengefasst lässt sich also festhalten, dass mit zunehmendem Strukturierungsgrad eines Textes auch dessen Eignung als KS-Objekt steigt. Ein Blick auf aktuelle KS-Software-Werkzeuge zeigt, dass die Art der Strukturierung unterschiedlich sein kann. Klassische Textverarbeitungssysteme unterstützen die Möglichkeit einer hierarchischen Textstruktur mit einer verschachtelten Unterteilung des Textes in Kapitel, Unterkapiteln und Abschnitten. Die vorliegende Dissertation ist ein Beispiel dafür.

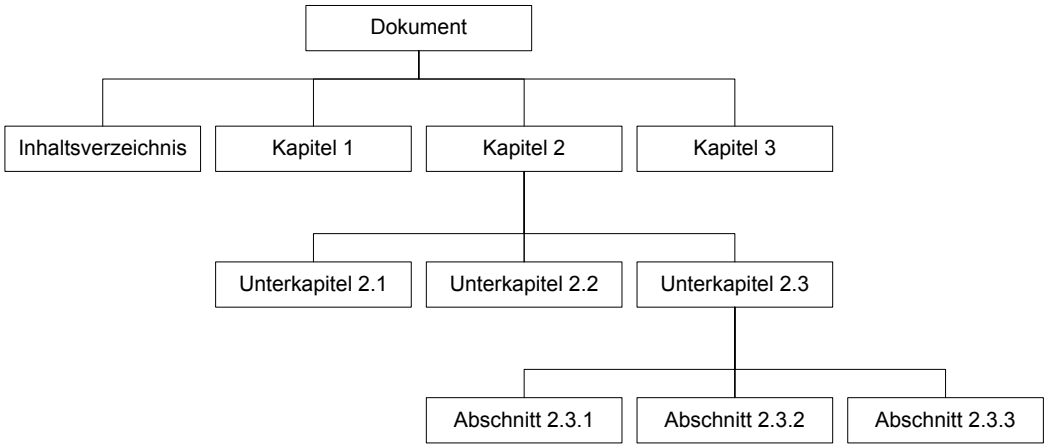

Abbildung 11: Beispiel für eine hierarchische Textstruktur

Eine vernetzte Textstruktur dagegen präsentiert sich dagegen als eine Sammlung gleichberechtigter Teiltexte, die über gegenseitige Verweise miteinander verbunden sind. Dies entspricht der Vernetzung von Artikeln in einem Lexikon oder einer Zeitung.

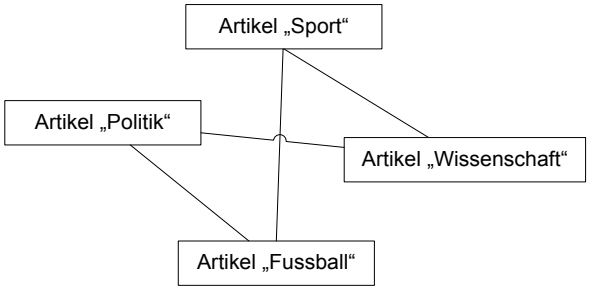

Abbildung 12: Beispiel für eine vernetzte Textstruktur

Die Grenzen zwischen hierarchischer und vernetzter Textstruktur sind fließend. Die Gliederung des Inhalts des Artikels kann durchaus wieder eine hierarchische Textstruktur beinhalten, die sich jedoch nicht dominierend auswirkt. Umgekehrt kann durch Setzen von Querverweisen eine hierarchische Textstruktur auch Ansätze einer vernetzten Textstruktur beinhalten.

Wie immer auch die Struktur eines Textes aussieht, je ausgeprägter sie ist, desto besser kann ein Text inhaltlich wie auch organisatorisch aufgeteilt und damit auf verschiedene Autoren verteilt werden.

4.3.3 Textverteilung

Steht die Textstruktur fest, kann die nächste Phase des KS-Projekts beginnen. Der Text muss unter den beteiligten Autoren aufgeteilt werden. Es bieten sich folgende Strategien an:

- Jeder Autor bekommt eine Kopie des Gesamttextes und arbeitet mit diesem weiter.

- Jeder Autor bekommt lediglich einen Teiltext und arbeitet nur an diesem Teiltext weiter.

- Im Extremfall bekommt jeder Autor seinen Teiltext exklusiv zugewiesen, d.h. nur er und keiner seiner Kollegen arbeiten an genau diesem Teiltext weiter.

Die Verteilung von Teiltexten hat zunächst den Vorteil, dass die Menge des Textes reduziert werden kann. Bei großen KS-Projekten mit sehr viel Text macht es wenig Sinn, allen Autoren stets den Gesamttext zuzuweisen. Dies würde unnötig viel Speicherplatz erfordern. Gleichzeitig wird aber auch die Konfliktgefahr erheblich reduziert. Ein Autor kann eben nur in seinem Teiltext Änderungen durchführen und nicht anderswo.

Auf der anderen Seite kann es jedoch wichtig sein den Überblick über den Gesamttext zu behalten. Das Verteilen von Gesamttextkopien hat zudem den Vorteil, dass auch schwache Textstrukturen sehr viel besser gemeinsam bearbeitet werden können. Gerade am Anfang des KS-Projekts, wo unter Umständen nur eine grobe Gliederung oder gar nur ein weißes Blatt Papier den Ausgangspunkt für die weitere Zusammenarbeit bildet, ist es wichtig, Einfluss auf den Gesamttext nehmen zu können. Das sich damit erhöhende Konfliktpotential lässt sich durch geschicktes Konflikt-Management eindämmen (siehe dazu Seite 58).

Reaktive KS-Systeme stellen den Autoren in der Regel den Gesamttext zur Verfügung. Jeder Autor kann zu jeder Zeit an jedem Abschnitt des Gesamttextes arbeiten. Parallele KS-Systeme, insbesondere jene, die auch Offline-Szenarien unterstützen, tendieren dazu mit Teiltexten zu arbeiten.

4.3.4 Textaktivitäten

Unter Textaktivitäten ist in dieser Arbeit, wie schon früher erwähnt, die eigentliche Arbeit des Schreibens mit allen unmittelbar damit zusammenhängenden Dingen wie Ideen sammeln und filtern und redigieren des Textes zu verstehen. Für das Redigieren des Textes, also das Nachprüfen und korrigieren haben sich die Änderungsverfolgung und Anmerkungen – jeweils mit Zeitstempel und Autor – als nützliche Instrumente in klassischer Textverarbeitungssoftware erwiesen. Im Hinblick auf die Erfordernisse eines KS-Systems wollen an dieser Stelle noch folgende weitere Funktionen ergänzen:

1. Das temporäre Sperren von Texten gegen Änderungen, z.B. durch den Lektor, der gerade diesen Text redigiert

2. Das endgültige Sperren von Texten gegen Änderungen, z.B. durch den Lektor, der diesen Text zur Veröffentlichung freigeben möchte.

3. Das Veröffentlichen von Texten, d.h. der Text wechselt seinen Status von „Entwurf“ zu „Veröffentlicht“.

4. Die Revisionierung von Texten, d.h. der Teiltext bzw. der Gesamttext enthält eine Versionsnummer mit Zeitstempel und entsprechenden Anmerkungen, um einen Zwischenstand festzuhalten, auf den ggf. später wieder Bezug genommen werden kann. Etwas Analoges kennt aus Werkzeugen zur kollaborativen Softwareentwicklung (z.B. Subversion [Collins-Sussman et al. 04]). Der Versionierung von Texten kann gleichfalls zur Konfliktvermeidung verwendet werden was im Folgenden noch näher erläutert wird.

4.3.5 Textsynchronisation

Durch Textverteilung wird jeder Autor in die Lage versetzt, seinen Beitrag zum Gesamttext beizutragen. Diese aus den Textaktivitäten resultierenden Änderungen eines jeden Autors müssen zu einem späteren Zeitpunkt wieder zusammengeführt werden, so dass ein neuer Gesamttext entstehen kann. KS-Systeme verwenden dafür zwei unterschiedliche Synchronisationstopologien, die unmittelbar mit der zugrundeliegenden System-Architektur verknüpft sind:

- ***n:m Synchronisationstopologie*:** Gleichberechtigte Peers gleichen direkt mit Echtzeitsynchronisation ihre Textkopien miteinander ab

- ***1:n Synchronisationstopologie*:** Clients arbeiten über einen ausgewiesenen Server miteinander zusammen, wobei die Textkopien mit dem Referenztext auf dem Server synchronisiert werden, was in der Regel mit Latenz, also nicht in Echtzeit erfolgt.

Die bei der Synchronisation möglicherweise entstehenden Konflikte müssen über das Konflikt-Management gelöst werden, auf das den im Folgenden noch detaillierter eingegangen wird.

4.3.5.1 n:m-Synchronisation

Bei einer n:m-Synchronisationstopologie besteht ein KS-System aus einer Peer-to-Peer-Architektur mit beliebig vielen gleichberechtigten KS-Peers. Jeder KS-Peer besitzt eine vollständige Kopie des Gesamttextes. Neue hinzukommende KS-Peer holen sich eine aktuelle Version des Gesamttextes von einem der bereits aktiven KS-Peer. Sobald ein KS-Peer Änderungen am Text vorgenommen hat, muss er diese Änderungen mit allen anderen aktiven KS-Peers synchronisieren.

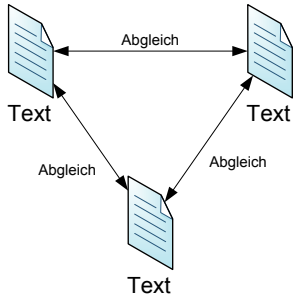

Abbildung 13: n:m-Synchronisationstopologie

Eine n:m-Topologie ist besonders gut für Ad-hoc-Kollaborationen (z.B. bei Meetings) mit einer überschaubaren Anzahl von Autoren geeignet.

Jeder KS-Peer muss alle anderen KS-Peers im Netzwerk kennen. Die Kommunikation ist multilateral. Dies erlaubt synchrones Arbeiten in Echtzeit. Textänderungen an einem KS-Peer können sofort an alle anderen KS-Peers übertragen und visualisiert werden. Es wird keine zentrale Infrastruktur in Form eines KS-Servers benötigt, was den Administrationsaufwand verringert. Fällt ein KS-Peer aus, können trotzdem alle anderen KS-Peers normal weiterarbeiten.

Der Nachteil der dezentralen Textsynchronisation liegt zum einem in der schnell steigenden Komplexität der Netzwerkkommunikation bei vielen Autoren (4 KS-Peers benötigen bereits 6 Kommunikationskanäle). Zum anderen sind die Autoren gezwungen synchron zu arbeiten.

4.3.5.2 1:n-Synchronisation

Bei einer 1:n-Synchronisationstopologie besteht ein KS-System aus einem ausgewiesenen KS-Server und beliebig vielen KS-Clients. Jeder KS-Client muss sich zunächst mit dem KS-Server verbinden, um die aktuelle zentral gespeicherte Version des gewünschten Textdokuments zu erhalten. Sobald ein KS-Client Änderungen am Text vorgenommen hat, kann er eine Synchronisation mit dem KS-Server vornehmen, um alle Änderungen in das zentrale Textdokument zu übertragen. Durch das erneute Abfragen des zentralen Textdokuments werden indirekt auch die Änderungen der anderen KS-Clients auf den eigenen KS-Client übertragen.

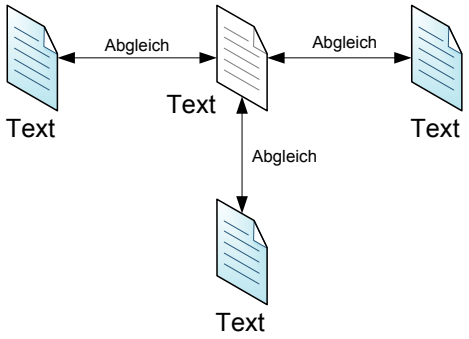

Abbildung 14: 1:n-Topologie

Diese Synchronisationstopologie wird von fast allen KS-Systemen implementiert. Die Vorteile liegen auf der Hand. Jeder KS-Client muss nur den KS-Server im Netzwerk kennen, eine direkte Kommunikation mit anderen KS-Clients ist nicht nötig. Dies erlaubt asynchrones Arbeiten. Ein Autor kann beispielsweise seine Textänderungen am Vormittag vornehmen und anschließend seine Arbeit beenden. Der nächste Autor, der erst am Nachmittag mit seiner Arbeit beginnt, erhält trotzdem eine aktuelle Version mit allen Textänderungen des ersten Autors, da dieser seinen Text mit dem KS-Server abgeglichen hat. Ein zentraler KS-Server bietet zudem erlaubt zudem eine zentrale Administration (z.B. Erstellen von Sicherungskopien und Statistiken oder die Weitergabe des Textes in der Publikationskette) des Gesamt-

textes. Eine 1:n-Topologie ist daher besonders gut für asynchrones Arbeiten und für eine hohe Anzahl an Autoren gedacht.

Der Nachteil eines KS-Servers liegt zum einem im erhöhten Administrationsaufwand. Der KS-Server muss in der Regel auf einem leistungsfähigen Computer mit 24x7-Stunden-Laufzeit installiert werden. Wird der Zugriff auf den KS-Server unterbrochen, hat dies Auswirkungen für alle KS-Clients (Single point of failure). Textänderungen können jetzt nicht mehr untereinander synchronisiert werden. Zum anderen wird durch die Latenz des Datenabgleichens (KS-Client gleicht mit KS-Server ab, danach propagiert KS-Server Änderungen an alle anderen KS-Clients) eine Textsynchronisation in Echtzeit praktisch unmöglich gemacht.

4.3.5.3 Hybride Topologien

Es können auch hybride Topologien aufgebaut werden, indem so genannte Cluster zu einer Hierarchie zusammengefasst werden. Jeder Cluster realisiert dabei eine der drei genannten Topologien und zwar unabhängig von den jeweiligen anderen Clustern.

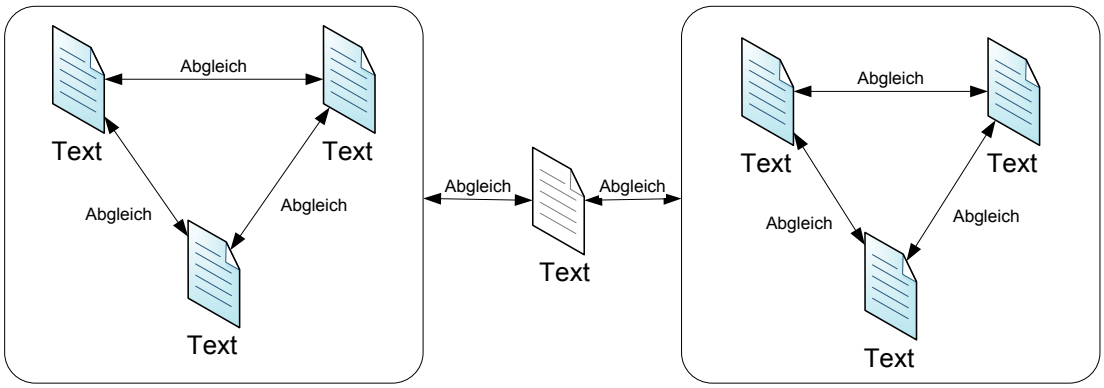

Abbildung 15: Hybride Topologie

Hier lassen sich wiederum unterschiedliche Strategien beobachten:

- Die veränderte Kopie des Gesamttexts wird mit dem Referenz-Gesamttext synchronisiert.
- Der komplette dem Autor zugewiesene Teiltext wird mit dem Referenz-Gesamttext synchronisiert.
- Nur die unmittelbar vom Autor durchgeführten Änderungen werden mit dem Referenz-Gesamttext synchronisiert.

4.4 Konflikt-Management

4.4.1 Konflikte

Beim Synchronisieren von Textänderungen kann es zu Konflikten kommen. Ein Konflikt ist ein Zustand, bei dem sich zwei oder mehrere Textänderungen gegenseitig ausschließen. Es lassen sich zwei Arten von Konflikten unterscheiden: Syntaktische Konflikte und semantische Konflikte.

Ein Beispiel für einen syntaktischen Konflikt:

> *Die Autoren A und B bearbeiten unabhängig voneinander die Gliederung eines gemeinsamen Textes. Autor A löscht einen Eintrag in der Gliederung, während Autor B den gleichen Eintrag an das Ende der Gliederung verschiebt. Beim Synchronisieren der Textänderungen von Autor A und Autor B kommt es zu einem Konflikt, da sich beide Änderungen gegenseitig ausschließen.*

Ein syntaktischer Konflikt entsteht genau dann, wenn zwei Autoren unabhängig voneinander ein und denselben Teiltext mit unterschiedlicher Intension verändern, und anschließend diese Textänderungen synchronisiert werden soll. Syntaktische Konflikte sind in der Regel leicht zu erkennen.

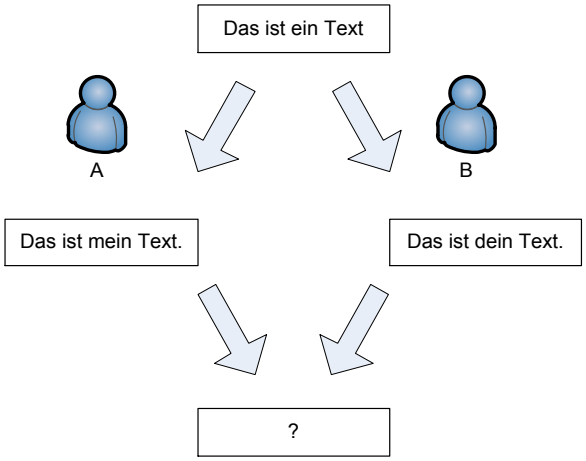

Abbildung 16: Syntaktischer Konflikt

Ein Beispiel für einen semantischen Konflikt:

> *Die Autoren A und B bearbeiten unabhängig voneinander die fünf-teilige Gliederung eines gemeinsamen Textes. Beide wollen die Gliederung etwas verkürzen. Autor A löscht die ersten zwei Einträge in der Gliederung, während Autor B die letzten zwei Einträge löscht. Beim Synchronisieren der Textänderungen von Autor A und Autor B kommt zu einem Konflikt, da der resultierende Text nur noch einen Gliederungspunkt enthält und somit viel zu kurz ist.*

Ein semantischer Konflikt entsteht genau dann, wenn zwei Autoren unabhängig voneinander Textänderungen durchführen, die bei einer anschließenden Synchroni-

sation zu keinem syntaktischen Konflikt führen, die Integrität des Textes aber dennoch verletzten. Semantische Konflikte sind in der Regel schwer zu erkennen.

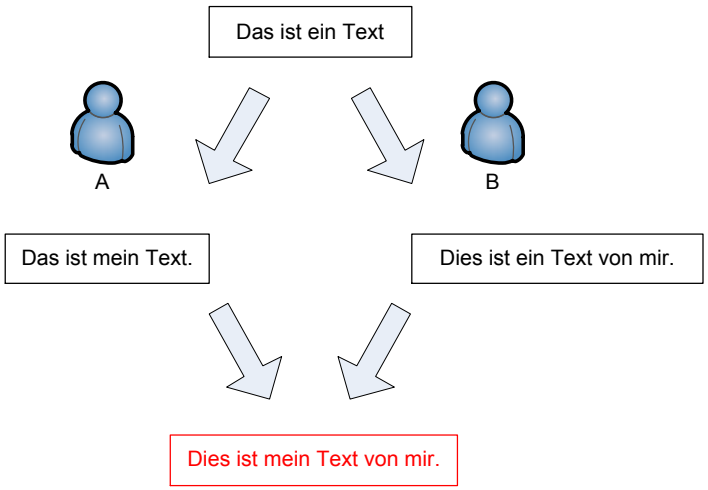

Abbildung 17: Semantischer Konflikt

4.4.2 Konflikterkennung

Bevor ein Konflikt aufgelöst werden kann, muss er erkannt werden. Der wichtigste Parameter ist dabei die Zeit. Sollen zwei Teiltexte miteinander synchronisiert werden, muss zunächst geklärt werden, wann der Teiltext zuletzt geändert wurde. Dafür wird zusätzlich ein Zeitstempel für diesen Teiltext hinterlegt. Mit Hilfe dieser beiden Angaben lassen sich Konflikte erkennen. Betrachtet man einen beliebigen Teiltext, so muss zunächst geklärt werden, wer diesen Teiltext zuletzt geändert hat. Hierfür wird jeder Teiltext explizit einem Autor zugewiesen wird. Der Autor selber wird durch einen eindeutigen Namen bzw. eine eindeutige Identifikationsnummer identifiziert. Ein Beispiel für eine syntaktische Konflikterkennung:

> *Die Autoren A und B bearbeiten unabhängig voneinander einen gemeinsamen Text. Beide ändern denselben Teiltext. Autor A löscht die ersten zwei Einträge in der Gliederung, während Autor B die letzten zwei Einträge löscht. Beim Synchronisieren der Textänderungen von Autor A und Autor B kommt es zu einem Konflikt, da der resultierende Text nur noch einen Gliederungspunkt enthält und somit viel zu kurz ist.*

In der Praxis müssen beide Angaben jedoch mit Vorsicht betrachtet werden, da sie nicht immer eindeutig sind. Melden sich beispielsweise zwei Autoren an unterschiedlichen Computern (ungewollt oder beabsichtigt) mit der gleichen Identifikation an, so kann später der Autor nicht unterschieden werden. Dies lässt sich durch zwei Maßnahmen unterbinden bzw. abmildern:

1. Ist ein Autor bereits mit einer bestimmten Identifikation im Textverarbeitungssystem angemeldet, so darf sich ein anderer Autor nicht mit der gleichen Identifikation anmelden. Das System gibt in diesem Fall eine Warnung heraus und benachrichtigt auf Wunsch den bereits angemeldeten Autor.

2. Kann eine Anmeldung mit gleicher Identifikation mehrfach erfolgen, so kann durch Hinzuziehen des Netzwerknamens oder der Netzwerkadresse des Computers ein weiteres Unterscheidungsmerkmal hinzugefügt werden. So kann bei gleicher Identifikation trotzdem erkannt werden, dass Änderungen von zwei unterschiedlichen Computern synchronisiert werden sollen.

Auch der Zeitstempel ist problematisch, da nicht garantiert werden kann, dass die Zeit auf zwei unterschiedlichen Computern auch wirklich gleich ist. Daher implementieren Textverarbeitungssysteme in der Regel eine Versionierung für Teiltexte. Dabei wird anstatt eines Zeitstempels eine Versionsnummer hinterlegt, die mit jeder Textänderung erhöht wird. Im Prinzip unterscheidet sich eine Versionsnummer nicht wesentlich von einem Zeitstempel, lediglich die Kalibrierung (Ausgangspunkt kann eine beliebige Zahl sein) und das Delta der Erhöhung (immer 1) ist verschieden.

4.4.3 Konfliktvermeidung durch Echtzeitsynchronisation

Bei einer Echtzeit-Synchronisation werden alle Änderungen unmittelbar und ohne Zeitverzögerung zwischen allen Autoren, die an unterschiedlichen Orten bzw. Clients arbeiten, abgeglichen.

In der Theorie ist eine Echtzeitsynchronisation frei von Konflikten, in der Praxis muss jedoch die Netzwerklatenz berücksichtigt werden. Das Übertragen der Daten im Netzwerk braucht nun einmal Zeit. Während dieser Zeit können Texte bereits ihren Inhalt erneut geändert haben, so dass das Einspielen der Änderungen zu einem inkonsistenten Zustand des Textes und somit doch zu einem Konflikt führen könnte. Dieses Problem lässt sich jedoch durch Technologien wie Operational Transformations [Sun et al. 06] weitestgehend beseitigen.

4.4.3.1 Operational Transformations (OT)

Operational Transformations (OT) ist eine Sammlung von Techniken bzw. Algorithmen für die Konsistenzerhaltung bei der Echtzeitsynchronisation. Ursprünglich 1989 von Ellis und Gibbs [Ellis et al. 89] beschrieben, wurden diese von anderen Autoren erweitert und diskutiert, insbesondere in der *Group of Collaborative Editing* (SIGCE, www.cit.griffith.edu.au/~scz/sigce) und den *Computer Supported Cooperative Work* Konferenzen (CSCW, www.cscw2010.org).

Bei OT werden die kausalen Abhängigkeiten zwischen Editieroperationen betrachtet, wobei die zeitliche Reihenfolge der Erzeugung der Operationen und deren Abarbeitungssequenz eine wesentliche Rolle spielt. Dazu ein einfaches Beispiel:

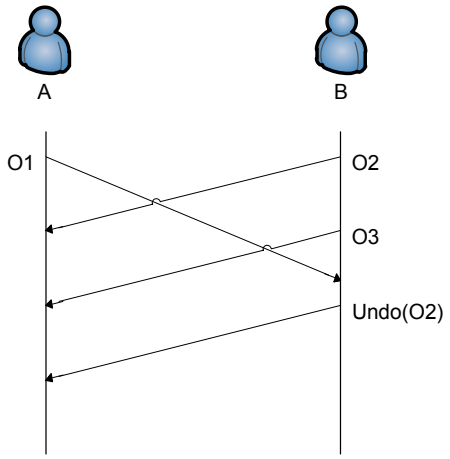

Abbildung 18: Beispiel für ein kollaboratives Echtzeitszenario

Ein Teiltext "abc" wird von zwei Autoren an zwei Clients[10] A und B mit Operationen O1, O2 und O3 bearbeitet, wobei der Autor an Client A die Operation O1 und der Autor an Client B die Operationen O2 und O3 ausführt:

1. O1 = Insert(0, "x"), Zeichen "x" an Position"0" einfügen
2. O2 = Insert(0, "y"), Zeichen "y" an Position"0" einfügen
3. O3 = Delete(2, "c"), Zeichen "c" an Position "2" löschen

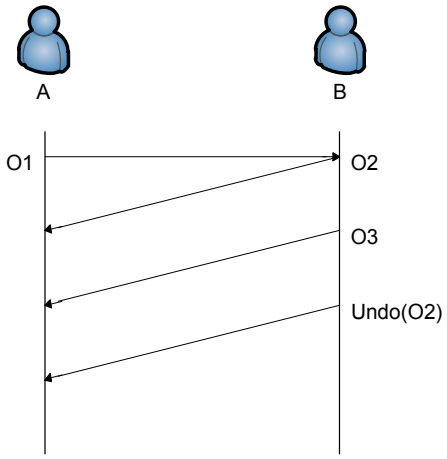

Abbildung 19: Alternative Operationsfolge

Da beide Autoren unabhängig voneinander arbeiten, ergibt sich ein Dilemma: Durch die Latenz im Netzwerk kann die Operation O1 an Client B eintreffen und ausgeführt werden (1) nachdem die Operation O2 dort ausgeführt wurde (wie in Abbildung 18)

[10] Wenn Autoren gemeint sind, die an unterschiedlichen Computern (Orten bzw. *Sites*) im Netz zusammenarbeiten, wird im Folgenden von *Clients* gesprochen, in der englischsprachigen Literatur wird in der Regel von *Sites* gesprochen.

oder (2) bevor die Operation O2 dort zur Ausführung ansteht (wie in Abbildung 19). Im Fall (1) wäre das der Teiltext „yxabc“, im Fall (2) wäre das Ergebnis „xyabc“.

Zeitpunkt	Kontext Client B	Ausgeführte Operation
t1	Abc	O1
t2	Xabc	O2
t3	Yxbc	O3
t4	Yxab	…

Tabelle 4: Client B Operationsfolge O1, O2, O3

Zeitpunkt	Kontext Client B	Ausgeführte Operation
t1	Abc	O2
t2	yabc	O3
t3	xybc	O1
t4	xyab	…

Tabelle 5: Client B Operationsfolge O2, O3, O1

Das Ergebnis hängt also davon ab, ob die Operation des jeweils anderen Autors vor oder nach der eigenen ersten Operation ausgeführt wird. Die Texte auf beiden Clients können somit divergieren.

Das Dilemma ist begründet in der örtlichen Unabhängigkeit der beiden Operationen. Die Lösung liegt darin, die auf jedem Client ausgeführten Operationen durch Transformationen bzw. explizite Festlegungen so zu korrigieren, dass insgesamt – also über alle Clients gesehen - Konvergenz erreicht wird. Unter *Konvergenz* verstehen wir, dass alle Dokumentkopien auf den Clients nach Ausführung einer identischen Menge von Operationen identisch sind. Dazu werden diverse Modelle vorgeschlagen, auf die im Folgenden eingegangen wird:

Alle Modelle implementieren im Endeffekt einen OT Algorithmus (oder Integrationsalgorithmus) der auf OT Funktionen und OT Eigenschaften basiert, für die es wiederum diverse Vorschläge gibt.

4.4.3.2 Konsistenzerhaltende Modelle

Konsistenzerhaltende Modelle versuchen die Konsistenz des Textes zu erhalten, indem sie kausale Abhängigkeiten betrachten. Wenn ein Ereignis E1 zeitlich vor einem Ereignis E2 stattgefunden hat, betrachtet man E2 als kausal abhängig von E1, wenn nicht, sind beide kausal unabhängig. Kausal unabhängige Ereignisse – oder in diesem Fall Textoperationen – können ohne Beachtung einer Reihenfolge auf dem jeweiligen Client ausgeführt werden. Das klappt aber gerade dann nicht, wenn die Operationen zwar kausal unabhängig sind, sich aber auf den gleichen Teiltext beziehen, wie gleich zu sehen ist.

Es soll nochmals das letzte Beispiel betrachtet werden. O1 ist kausal unabhängig von O2 und O3, weil die letzten beiden Operationen von einem anderen Autor (an einem anderen Client) erzeugt wurden. O3 ist kausal abhängig von O2, da diese beiden Operationen von einem Autor zeitlich nacheinander erzeugt wurden. Entsprechend sind die beiden Relationen „Kausal abhängig“ und „Kausal unabhängig“ definiert:

Definition (Kausal abhängig)

> Für zwei Operationen O1 und O2, die jeweils von zwei Clients erzeugt wurden, gilt: O2 ist kausal abhängig von O1, notiert als $O1 \rightarrow O2$, wenn (1) derselbe Autor O1 vor O2 generiert oder wenn (2) bei unterschiedlichen Autoren A und B die Ausführung von O1 vor der Generierung von O2 stattfindet oder wenn (3) eine Operation O3 existiert, so dass $O1 \rightarrow O3$ und $O2 \rightarrow O2$ gilt.

Definition (Kausal unabhängig)

> Für zwei Operationen O1 und O2, die jeweils von zwei Clients erzeugt wurden, gilt: O2 ist kausal unabhängig von O1, notiert als $O1||O2$, wenn weder $O1 \rightarrow O2$ noch $O2 \rightarrow O1$ gilt.

Ein Ziel im Rahmen von OT ist es, zwei Textoperationen, die auf einem Teiltext ausgeführt werden sollen, sinnvoll zu einer neuen Operation zu *transformieren*, die den Effekt beider Einzeloperationen enthält:

Definition (Inclusion Transformation Function IT)

> Die Funktion IT(O1, O2) = O sei die Funktion, die die Operation O1 gegen die Operation O2 so transformiert, dass der Effekt von O1 und O2 in der resultierenden Operation O enthalten ist.

Ein Beispiel für eine solche Transformation (OT-Funktion), die aus den zwei Operationen bzw. Transformationen O1 (erzeugt an Client A) und O2 (erzeugt an Client B) eine konkretes Resultat erzeugt, ist z.B. die folgende Funktion IT:

Beispiel: Funktion T

```
T(Insert(p1, z1, A), Insert(p2, z2, B)):=
 if (p1 < p2) return Insert(p1, z1, A) else
 if (p1 = p2 and A < B) return Insert(p1, z1, A)
 else return Insert(p1 + 1, z1, A)
```

Dieses Beispiel beschreibt nur ein Teil der Gesamtfunktion. Besteht die Menge primitive Operationen aus *Insert*, *Delete* und *Update*, dann müssen alle möglichen Kombinations-Tupel, d.h. konkret sechs, definiert werden. Zu dieser Funktion invers wäre die Funktion $inv\ T$, welche diese Transformation wieder rückgängig macht, dann so definiert:

Beispiel: Funktion inv(T)

```
inv(T(Insert(p1, c1, A),Insert(p2, c2, B)) :=
 if (p1 < p2) return Insert(p1, c1,A)
 else if (p1 = p2) and (A < B) return insert(p1, c1 ,A)
 else return ins(p1 – 1, c1, A)
```

Außerdem soll die Funktion org eingeführt werden, welche die Originaloperation am Anfang einer Operationskette zurückliefert, also $org\ O\ = O$, falls es sich bei O um eine Originaloperation handelt.

Das Grundmodell geht davon aus, dass wie oben erwähnt kausal unabhängige Operationen auf Clientseite in beliebiger Reihenfolge ausgeführt werden können, was aber in vorherigen Beispiel zum bekannten Dilemma führt. Die Lösung des oben genannten Dilemmas besteht darin, die Operationen nicht direkt auszuführen, sondern deren Ausführung an weitere konvergenzerhaltende Bedingungen zu knüpfen. In dieser Hinsicht macht es durchaus Sinn, die Ausführung einer Operation an den Kontext zu knüpfen, in dem sie erzeugt wurde. In vorherigen Beispiel sind die Operationen O1 und O2 basieren auf demselben gemeinsamen Teiltext erzeugt worden. Sie sind also in demselben Kontext C erzeugt worden: $C(O1)\ =\ C(O2)$. $C(O3)$ ist ungleich zu $C(O1)$ wie auch zu $C(O2)$, da O3 nach O2 erzeugt wurde. Wenn O2 bei Client A1 zur Ausführung ankommt, entspricht der $C(O2)$ nicht mehr dem Kontext auf diesem Client, da dort bereits O1 ausgeführt wurde und sich somit der Zustand des Textes also der Kontext geändert hat. Der Kontext einer Operation lässt sich z.B. als Mengenoperation beschreiben:

Definition (Kontext einer Operation)

1. Für eine originale Operation O gilt C(O) = DS, wobei DS den Zustand des Dokuments (*document state*) darstellt, zu dem O generiert wurde.

2. Für eine inverse Operation $Inv(O)$ gilt $C(inv(O) = C\ O\ \cup \{O\}$, wobei O die Funktion ist die rückgängig gemacht werden soll.

3. Für den Kontext einer transformierten Operation O‘ gilt: $C(O') = C\ O\ \cup \{org(Ox)\}$, wobei $O' = IT(O, O_x)\}$.

In diesem Beispiel ergibt sich $C\ O1\ = \{\}, C\ O2\ = \{\}$ und $C\ O3\ = \{O2\}$. Die transformierte Operation $O'2 = IT\ O2, O1$ hat dann den Kontext $C(O'2)\ = C\ O'2\ = \{O1\}$.

Ein möglicher Algorithmus, um das Dilemma aus Abbildung 18 zu lösen, wäre folgender:

Algorithmus (MatchContext)

Überprüfe für jede an einem Client mit Kontext C zur Ausführung anstehende Operation O mit Originalkontext C(O), ob dieser Kontext mit dem aktuellen oder einem früheren Kontext auf dem Client übereinstimmt:

1. *Ist dies der Fall, führe die Operation O auf dem Client aus.*
2. *Ist dies nicht der Fall, mache die Operationen* $\{O1 \ldots On\}$ *auf dem Client rückgängig, bis der Kontext C(O) oder ein Originalkontext erreicht ist. Anschließend transformiere die Operationen* $O, O1 \ldots On$ *mit der Funktion T und führe die transformierte Operation O' auf dem Kontext C(O) aus.*

Die Arbeitsweise des Algorithmus soll nun anhand des Beispiels betrachtet werden, bei dem vorausgesetzt wird, dass C(O1) = C(O2):

Zeitpunkt	Kontext Client A	Auszuführende Operation	Ausgeführte Operation	Bemerkung
t11	abc	O1	O1	abc=C(O1) = C(O2)
t12	xabc	O2	inv(O1)	xabc ≠ C(O2)
t13	Abc	T(O1, O2)	T(O1, O2)	abc=C(O2)
t14	xyabc	O3	Inv (T(O1, O2))	xyabc ≠ C(O3) = yabc
t15	Abc	T(T(O1, O2), O3)	T(T(O1, O2), O3)	Originalkontext
t16	xyb	…	…	…

Tabelle 6: Operationsfolge Client A

Zeitpunkt	Kontext Client B	Auszuführende Operation	Ausgeführte Operation	Bemerkung
t21	abc	O2	O2	
t22	yabc	O3	O3	yabc = C(O3)
t23	yab	O1	inv(O3), inv(O2)	C(O1)=xabc ≠ yab
t24	abc	T(T(O1, O2), O3)	T(T(O1, O2), O3)	Originalkontext
t25	yabc	…	…	…

Tabelle 7: Client B Operationsfolge O2, O3, O1

Wenn nun einmal alternativ angenommen wird, dass die Operationsfolge an Client B wie oben schon diskutiert O1, O2, O3 beträgt, erhält man mit dem Algorithmus MatchContext dasselbe Ergebnis, d.h. die Texte auf beiden Clients konvertieren in beiden Fällen:

Zeitpunkt	Kontext Client B	Auszuführende Operation	Ausgeführte Operation	Bemerkung
t21	abc	O1	O1	
t22	Xabc	O2	inv(O1)	xabc ≠ C(O2)
t23	Xabc	T(O1, O2)	T(O1, O2)	
t24	xyabc	O3	inv(T(O1, O2))	C(O3)=yabc ≠ xyab
t25	abc	T(T(O1, O2), O3)	T(T(O1, O2), O3)	Originalkontext
t25	xyab	…	…	…

Tabelle 8: Client B Operationsfolge O1, O2, O3

Konsistenzerhaltende Modelle fokussieren auf elementarer Operationen, genauer gesagt bestimmen sie, in welcher Reihenfolge welche Operationen transformiert werden sollen. Im Zweifelsfall bedeutet dies, die bisherigen Operationen bzw. Transformationen wieder rückgängig zu machen, um einen konsistenten Ausgangspunkt für Ausführung einer neu angekommenen Operation zu finden. Das kann sehr ineffizient sein. Entscheidend für die Effizienz bzw. Komplexität des Algorithmus *MatchContext* sind der Umfang des Teiltextes und die Anzahl der unabhängigen Operationen. Die Komplexität des Algorithmus ließe sich verringern, wen der Umfang des Teiltextes bzw. der Teiltexte verringert wird. Wenn in Betracht gezogen wird, dass ein Dokument nicht nur eine lineare Folge von Zeichen bzw. Wörtern ist, können die semantische Struktur des Dokuments verwendet werden, um die Aufteilung des Gesamtdokumentes in Teiltexte entsprechen der semantischen Struktur vorzunehmen. In den folgenden beiden Kapiteln werden Modelle vorgestellt, die Informationen auf höheren Abstraktionsebenen mit einbeziehen, um das OT-System effizienter zu machen.

Im Beispiel dieses Kapitels basierte das Modell auf den beiden Prinzipien

- Kausalität (**C**ausality) und
- Konvergenz (**C**onvergence)

Entsprechend werden Modelle, die auf diesen beiden Prinzipien Konsistenzerhaltung und Konvergenzerhaltung beruhen, in der Literatur als *CC-Modelle* kategorisiert. Es wurden auch Modell vorgeschlagen, die das Konzept der der Konvergenz weniger strikt sehen, indem sie als zusätzliche Bedingung die Intentionserhaltung (*intention preservation*) in das Modell mit aufnehmen. Als Illustration dieses Prinzips wird in der in der Literatur häufig folgendes Beispiel diskutiert:

Beispiel

> Ausgangspunkt sind drei Clients mit dem Starttext „abc“. Folgende Operationen werden nebenläufig auf den Client erzeugt:
>
> Insert(2, "x", Client1) - auf Client1 das Zeichen "x" nach "b" einfügen

Insert(1, "y", Client2) - auf Client2 das Zeichen "y" nach "a" einfügen

delete(1, Client3) - auf Client3 das Zeichen "b" löschen

Intuitiv müsste „ayxc“ das korrekte Resultat sein. Trotzdem wird „axyc“ ebenfalls als korrektes Resultat zugelassen.

Analog dazu kann man auch das zuvor erwähntes Beispiel als weitere Illustration des Intentionsprinzips heranziehen. In diesem Beispiel wurde erwähnt, dass bedingt durch die Latenz im Netz für Client B sowohl die Operationsfolge O1, O2, O3 wie auch O2, O3, O1 zur Ausführung anstehen könnte (siehe Tabelle 4 und Tabelle 5). Durch entsprechende Bedingungen wurde die erste Folge als korrekt bestimmt. Nun soll das Modell dahingehend erweitert werden, dass beide Ausführungsfolgen der Intention (der Autoren) entsprechen, es also sowohl das Resultat „yxab“ als auch das Resultat „xyab“ akzeptiert.

Die *Intention einer Operation O* wird dabei definiert als der Ausführungseffekt der Operation auf den Kontext C(O), in dem die Operation definiert wurde. Modelle, die das Intentionsprinzip als dritte Bedingung enthalten, werden in der Literatur unter *CCI-Modelle* (**C**onsistency, **C**onvergence, **I**ntention) zusammengefasst.

Die sogenannten *CSM-Modelle* (siehe z.B. [Rui Li, Du Li 2007]) versuchen das schwer zu formalisierende und damit schwer beweisbare Intuitionskonzept durch eine Erweiterung der Operationsrelation zu ersetzen. CSM-Modellen basieren auf den folgenden Prinzipien:

- Kausalität ***(Causality)***
- ***Single-operation effects***: Der Effekt der Operation O nach Ausführung ist derselbe Effekt wie der Effekt der Operation im Erzeugungskontext C(O).
- ***Multi-operation effects***: Die Effektrelation von zwei Operationen wird nach Ausführung beider Operationen weitergeführt.

4.4.3.3 OT-Daten und Operationsmodelle

Ein OT-System basiert genauer gesagt auf zwei Modellen: dem Datenmodell und dem Operationsmodell. Das Datenmodell repräsentiert das Dokument, bestehend aus seiner Struktur (z.B. linear oder hierarchisch) und den Objekten bzw. Daten. Das Operationsmodell legt fest, welche Operationen zur Verfügung stehen (z.B. Insert und Delete) und mit Hilfe welcher IT-Funktionen diese angewendet werden können. Bei genauerer Betrachtung der Struktur des OT-Systems, kann man innerhalb des Operationsmodells folgende Komponenten unterscheiden:

- Den OT-Algorithmus, der bestimmt welche Operationen gen welche transformiert wird.

- Die OT-Transformationsprozeduren, die bestimmen wie ein Tupel aus zwei elementaren Operationen gegeneinander transformiert werden kann

- OT-Bedingungen und OT-Eigenschaften, die das Verhalten des OT-Algorithmus und der OT-Transformationen bestimmen.

Durch die Trennung in den OT-Algorithmus auf höhere Abstraktionsebene von den OT-Transformationen auf der niederen Abstraktionsebene wurden einerseits generische OT-Algorithmen vorgeschlagen die auf unterschiedlichen Datenmodellen arbeiten können wie auch OT-Bedingungen und OT-Eigenschaften, die keine totale Ordnung der Operationsausführung mehr erfordert. Was das Operationsmodell angeht, kann man zwei Ansätze unterscheiden:

1. Generische Operationen: Diese Ansätze arbeiten mit Operationen auf höherer Abstraktionsebene bzw. auf Applikationsebene. Für diese Operationen werden Transformationen angegeben, die es erlauben diese Operationen generisch aus drei elementaren Operationen Insert, Delete und Update abzuleiten. Die Transformationsfunktionen können dabei in verschiedenen Anwendungsbereichen angewendet werden.

2. Applikationsspezifische Operationen: Die Operationen auf Applikationsebene ergeben sich aus der Kombination der elementaren Operationen wie sie bereits im vorletzten Kapitel bei der Funktion *T* zu sehen waren. Bei n verschiedenen elementaren Operationen werden dann n x n Transformationsfunktionen benötigt. Diese Transformationsfunktionen sind applikationsspezifisch und können nicht in anderen Anwendungen verwendet werden.

So wird in [Ignat et al. 03] das Dokument über eine Baumstruktur repräsentiert, auf der analog die im letzten Kapitel beschriebenen Funktionen rekursiv angewendet werden. Die Baumstruktur bildet folgende Granularitätsebenen entsprechend der syntaktischen Elemente in natürlicher Sprache ab:

(1) Dokument, (2) Paragraf, (3) Satz, (4) Wort, (5) Zeichen

Entsprechend der Baumstruktur werden Operationen definiert, die sich durch folgendes n-Tupel beschreiben lassen:

<level, type, position, content, stateVector, initiator>

wobei

level ∈ {1,2,3,4} die Granularitätsebene angibt

type ∈ {Insert, Delete} den Typ der Operation angibt

position ∈ {1… n} ein Vektor aus Positionsanagaben ist und *position[i]* die Position auf dem i-ten Granularitätslevel darstellt

content den Knoten repräsentiert, der den Inhalt der Operation beschreibt

stateVector der Zustandsvektor der die Operation generierenden Site ist

initiator die Identifikation (ID) der initiierenden Site darstellt

Eine Einfüge-Operation ist dann zum Beispiel *InsertWord(3, 1 , 2, "Baum")*, d.h. das Wort „Baum" wird in Absatz 3 in Satz 1 an Position 2 eingefügt.

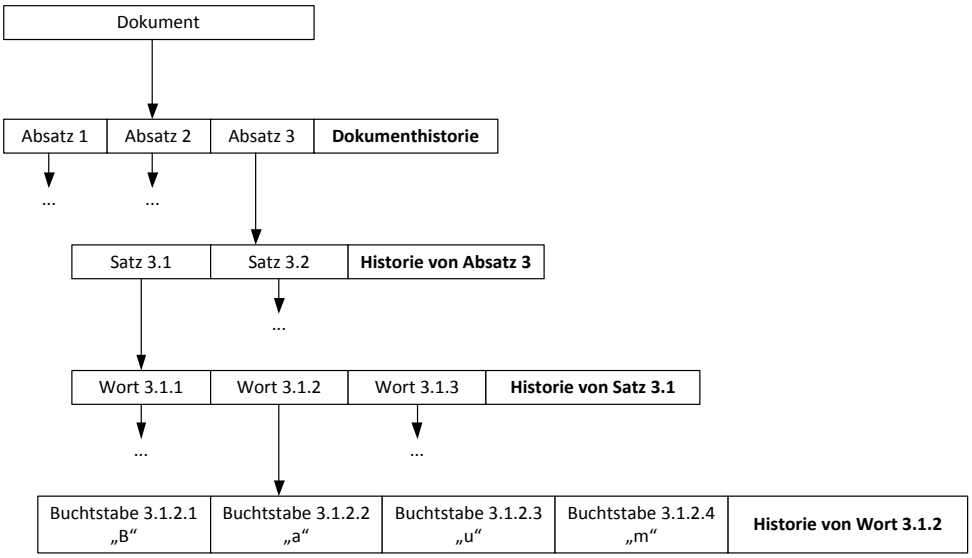

Abbildung 20: Text als Baumstruktur

Diese Baumstruktur erlaubt es, auf jeder Granularitätsebene den Algorithmus *MatchContext* rekursiv anzuwenden. Eine detaillierte Beschreibung eines solchen Algorithmus ist in [Ignat et al. 03] zu finden.

4.4.4 Konfliktvermeidung durch pessimistisches Sperren

Durch pessimistisches Sperren wird einem Autor exklusiver Schreibzugriff auf einen Text oder einen Teiltext gewährt. Andere Gruppenmitglieder können zur gleichen Zeit nur lesen. Erst nach Aufheben der Sperre kann ein anderer Autor wiederum exklusiven Schreibzugriff erlangen.

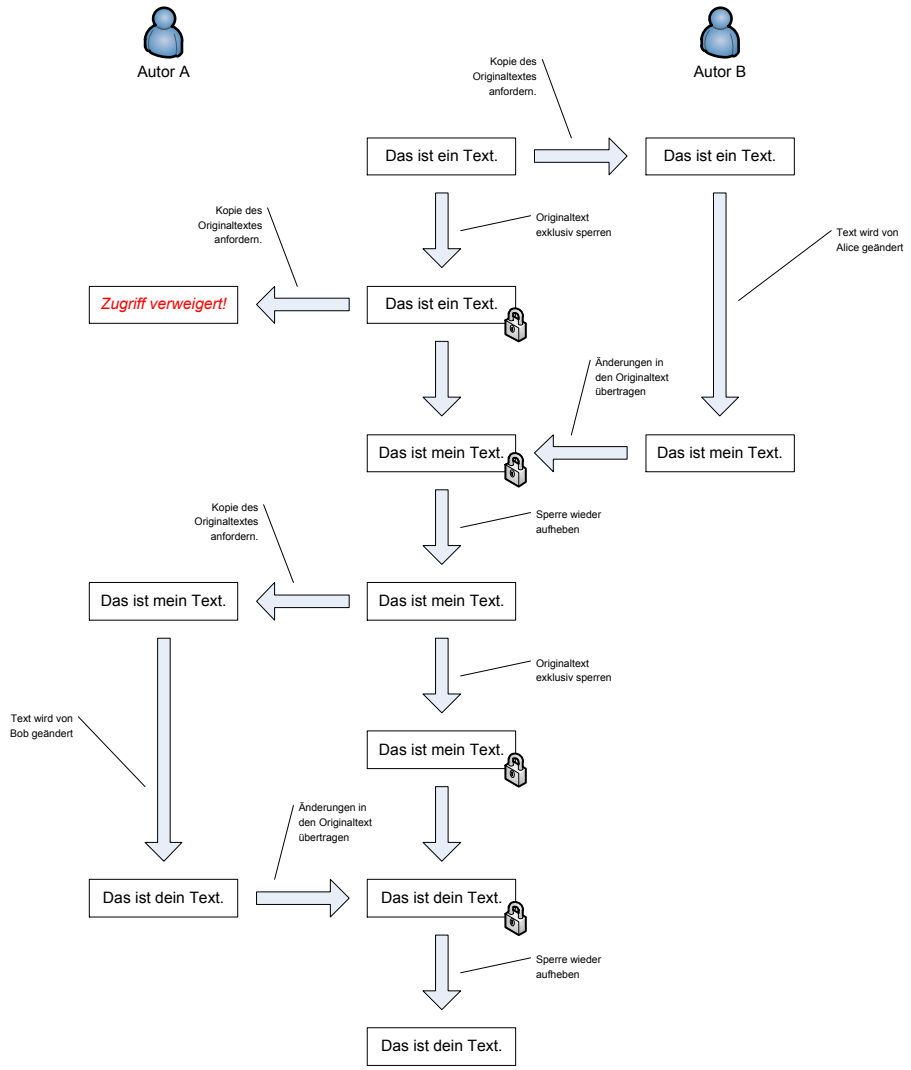

Abbildung 21: Pessimistisches Sperren

Durch pessimistisches Sperren können Konflikte von vorneherein ausgeschlossen werden, da zu einem Zeitpunkt nie mehr als ein Autor schreiben darf. Allerdings kann die Zusammenarbeit darunter leiden. Je nachdem wie viel Text gesperrt wird, müssen andere Autoren warten, bis sie weiterschreiben können. Dies lässt sich abmildern, indem Sperren nur auf einzelne möglichst feingranulare Abschnitte gelegt werden und nicht etwa auf den Gesamttext. Ein anderes Problem kann entstehen, wenn eine Sperre über einen langen Zeitraum fortbesteht (z.B. weil ein Autor gewollt oder ungewollt die Sperre nicht aufgehoben hat). Dies kann ebenfalls die Zusammenarbeit einschränken. Eine Lösung für dieses Problem könnte darin bestehen, Sperren mit einem Timeout zu belegen, so dass sie sich nach Ablauf einer bestimmten Zeitspanne automatisch wieder entsperren. Pessimistisches Sperren ist sehr restriktiv, vermeidet dafür aber logische Konflikte und ist daher für eine Kollaboration mit potentiell hohem Konfliktaufkommen geeignet.

4.4.5 Konfliktvermeidung durch optimistisches Sperren

Durch optimistisches Sperren wird einem Autor nicht-exklusiven Schreibzugriff auf einen Text oder Teiltext gewährt. Erst beim Synchronisieren seiner Änderungen mit dem Gesamttext wird überprüft, ob in der Zwischenzeit schon ein anderer Autor den gleichen Text oder Teiltext geändert hat. In diesem Fall entsteht ein Konflikt, der aufgelöst werden muss.

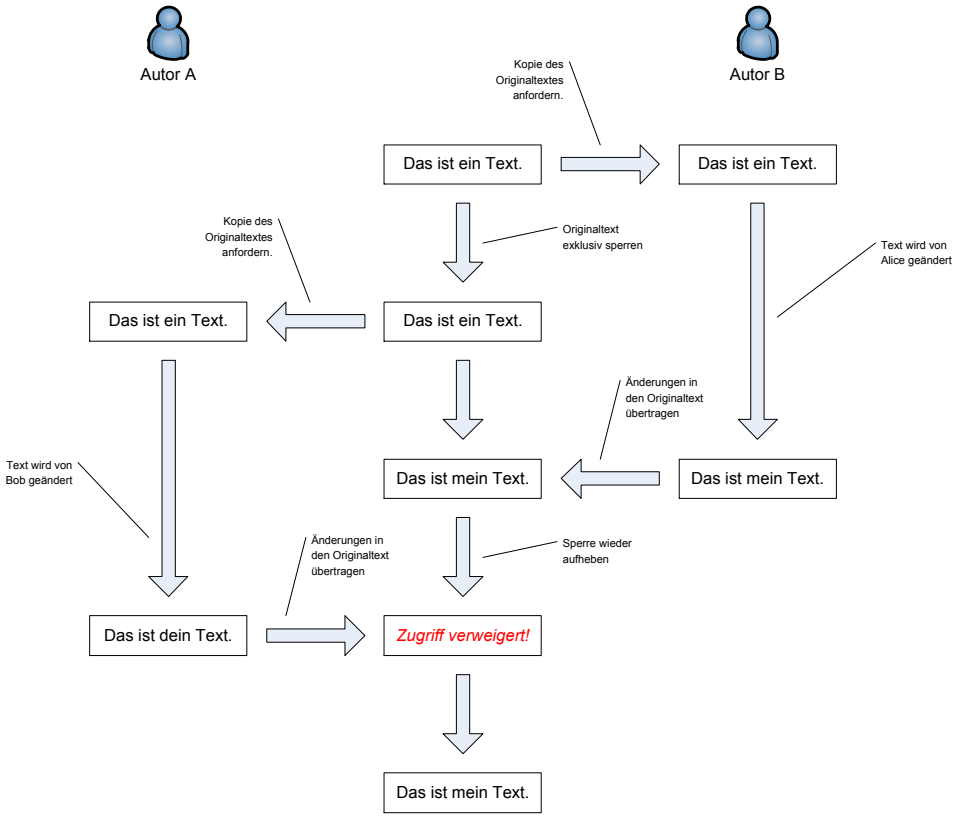

Abbildung 22: Optimistisches Sperren

Durch optimistisches Sperren werden mögliche Konflikte in Kauf genommen. Dafür wird die Zusammenarbeit kaum eingeschränkt. Mehrere Autoren können gleichzeitig am selben Text arbeiten und kein Autor wird durch eine Sperre blockiert. Um das Konfliktaufkommen jedoch möglichst gering zu halten, müssen sich die Autoren disziplinieren und besser untereinander absprechen. Optimistisches Sperren macht nur dann Sinn, wenn das zu erwartende Konfliktpotential eher gering ist.

4.4.6 Konfliktvermeidung durch explizites Sperren

Konfliktvermeidung durch optimistisches oder pessimistisches Sperren hat stets den Nachteil, dass es zu einer First-Come-First-Win-Situation kommen kann, d.h. der Autor, der als erster auf einem Text zugreift hat einen strategischen Vorteil gegenüber dem Autor, der nach ihm dasselbe vorhat. Eine Lösung dieses Problems besteht darin, Texte bzw. Teiltexte explizit mit Zugriffsrechten zu versehen, so dass unabhängig vom Zeitpunkt immer nur derjenige Autor Zugriff bekommt, welcher im Moment für diesen Text verantwortlich ist.

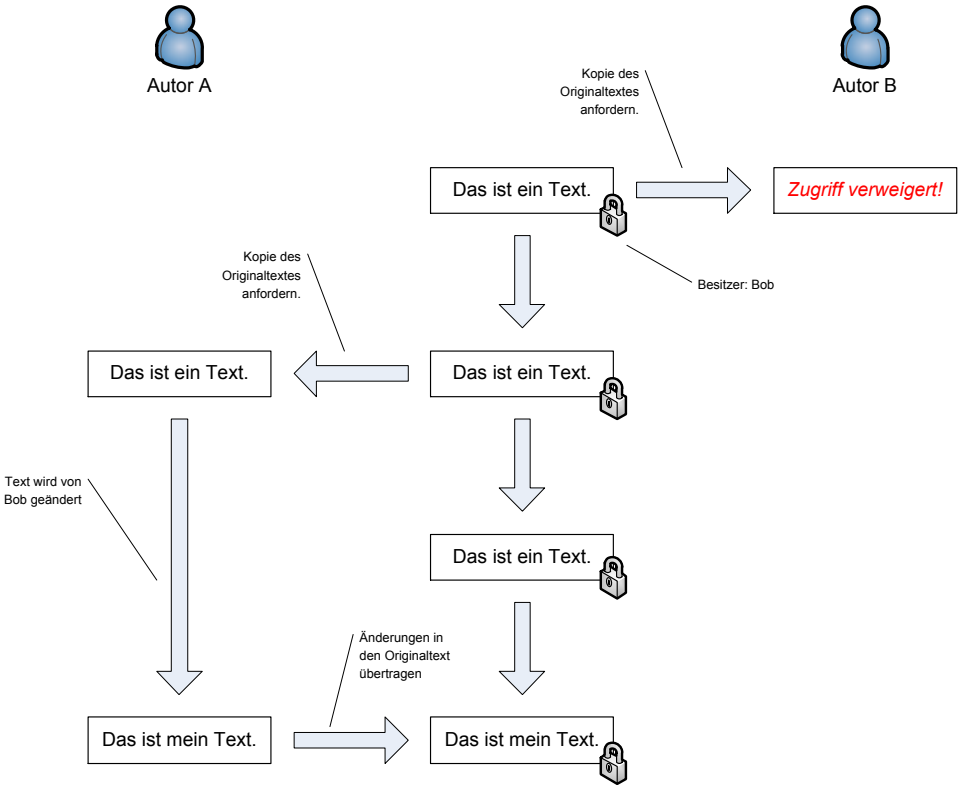

Abbildung 23: Explizites Sperren

Die Effektivität dieses Ansatzes hängt wesentlich von der Granularität der möglichen Rechtevergabe ab. So macht das Sperren des gesamten Textes, wie es beispielsweise im Kontext von Dokumenten-Management-Systemen praktiziert wird, wenig Sinn, möchte man eine parallele Texterstellung unterstützen. Vielmehr wäre eine Rechtevergabe auf Kapitelebene oder besser noch auf Abschnittsebene beliebiger Granularität (z.B. Unterkapitel, Abschnitte oder Satzfolgen) wünschenswert.

4.4.7 Weiterer Möglickeiten der Konfliktauflösung

Bei einer automatischen Konfliktauflösung durch Prioritäten werden zwei konkurrierende Textänderungen daraufhin untersucht, ob einem der beiden Autoren eine höhere Priorität als dem anderen Autor zugeordnet wurde. Der Autor mit der höheren Priorität gewinnt und seine Textänderung wird übernommen.

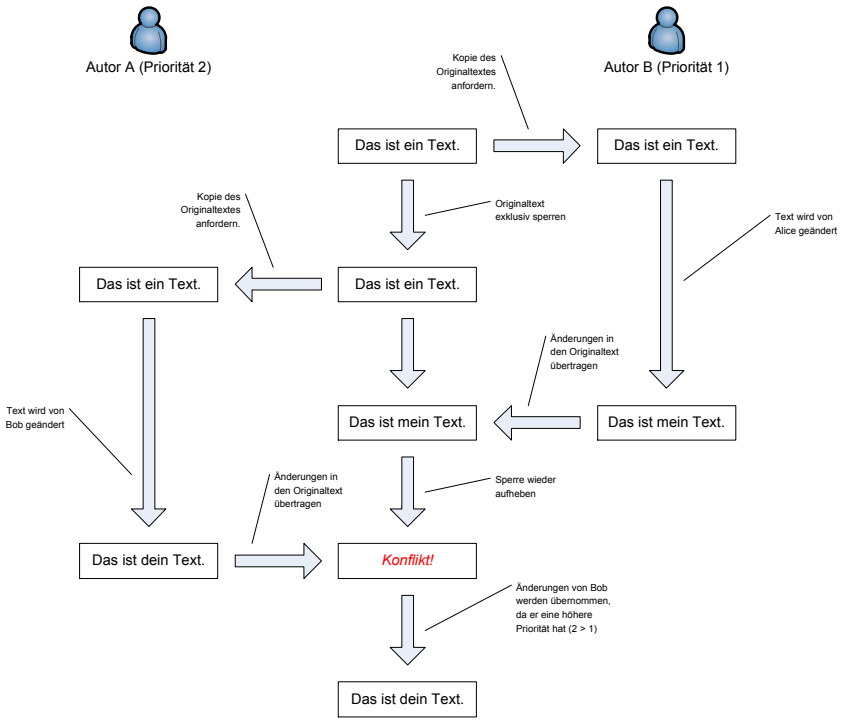

Abbildung 24: Konfliktauflösung durch Prioritäten

Bei einer automatischen Auflösung durch Vereinigung wird versucht, zwei konkurrierende Teiltexte zu einem Teiltext zu vereinigen. Dies funktioniert in der Regel jedoch nur, wenn sich die Änderungen nicht auf den eigentlichen Textinhalt beziehen (z.B. Layout-Änderungen, Rechtschreibung, etc.).

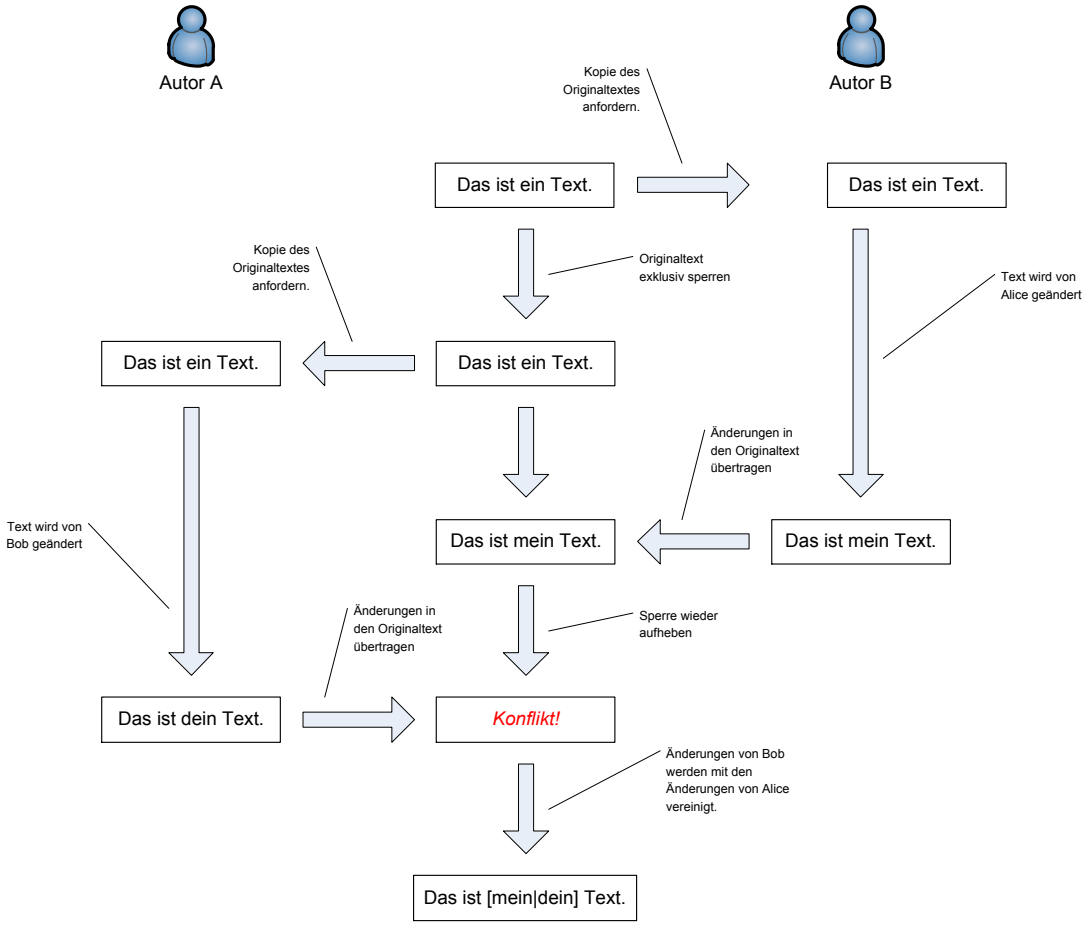

Abbildung 25: Konfliktauflösung durch Vereinigung

Bei einer manuellen Konfliktauflösung müssen die Teammitglieder sich untereinander absprechen, welche Änderungen übernommen und welche verworfen werden. Eine manuelle Auflösung ist immer dann notwendig, wenn alle anderen Auflösungsstrategien verworfen werden müssen.

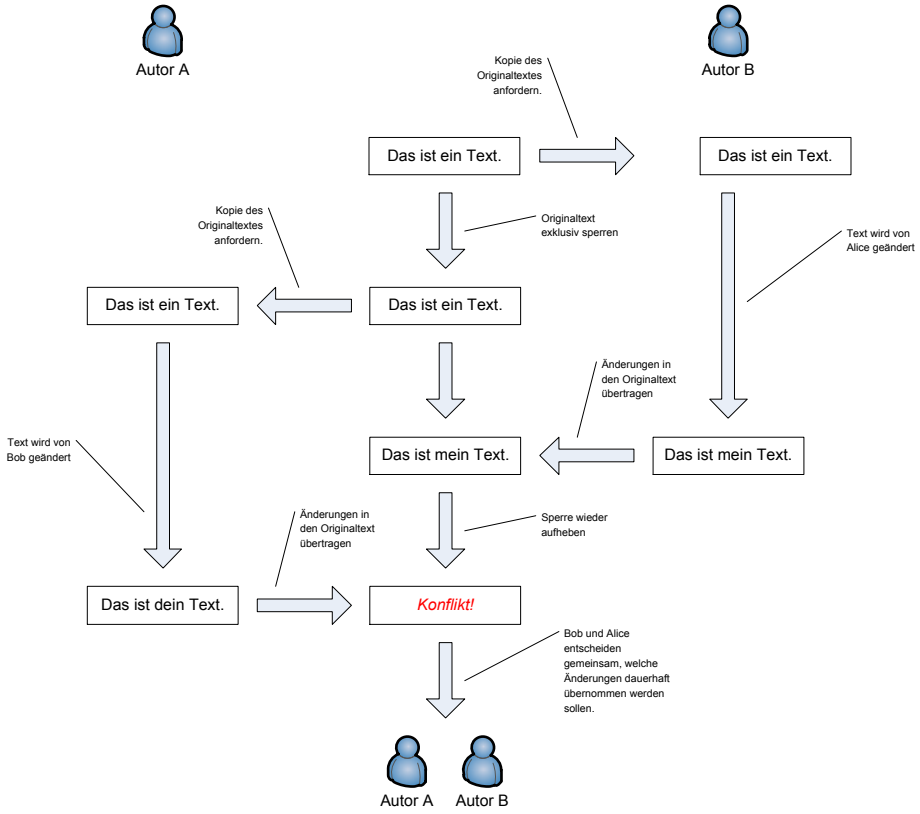

Abbildung 26: Manuelle Konfliktauflösung

4.5 Nachvollziehbarkeit

Durch die Nachvollziehbarkeit von Textänderungen kann der kooperative Erstellungsprozess eines Textes besser kontrolliert werden. Die gebräuchlichsten Entwurfsmuster in diesem Zusammenhang sind:

- Nachvollziehbarkeit durch Aktionsaufzeichnung
- Nachvollziehbarkeit durch Textversionierung

4.5.1 Aktionsaufzeichnung

Bei einer Aktionsaufzeichnung werden alle getätigten Eingaben in Form eines Aktionslogbuchs mitgeführt. In der Regel wird dabei zwischen drei Aktionen unterschieden: Einfügen von Text, Löschen von Text sowie Formatierung (z.B. Schriftart oder Schriftfarbe) des Textes. Der folgende Ausschnitt aus einem Open Office-Dokument zeigt die Aktionsaufzeichnung einer Texteinfügung:

```
<text:tracked-changes>
 <text:changed-region text:id="c001">
 <text:insertion>
  <office:change-info>
  <dc:creator>Frank Stüber</dc:creator>
  <dc:date>2010-06-18T12:56:04</dc:date>
  </office:change-info>
 </text:insertion>
```

```
 </text:changed-region>
</text:tracked-changes>

<text:p>
 Das ist der Originaltext<text:change-start text:change-id="c001"/>,
 und dies wurde hinzugefügt<text:change-end text:change-id="c001"/>.
</text:p>
```

Zu sehen ist kleiner Textausschnitt („Das ist der Originaltext.“), dessen Inhalt zu einem späteren Zeitpunkt (am 18.6.2010 um 12:56) vom Autor Frank Stüber um einen weiteren Teilsatz („und dies wurde hinzugefügt“) ergänzt wurde. Durch Rückanwendung der Texteinfügung lässt sich aus dem aktuellen Text wieder der ursprüngliche Text herstellen. Ein Autor bekommt durch diese Aktionsaufzeichnung folgende Informationen geliefert:

- Welcher Autor hat wann welche Änderungen an einem Text durchgeführt?
- Wie sah der Text vor diesen Änderungen aus?

Ein Nachteil der Aktionsaufzeichnung ist, dass bei häufigen und umfangreichen Änderungen die Größe des Logbuches schnell anwachsen kann. Dies kann sogar dazu führen, dass das Logbuch mehr Speicherplatz verbraucht als der eigentliche Text. Eine Möglichkeit der Optimierung besteht darin, das Logbuch nur bis zu einem bestimmten Zeitpunkt in der Vergangenheit mitzuführen (z.B. nur Änderungen innerhalb der letzten 14 Tage). Eine weitere Möglichkeit besteht in der Umstellung auf eine Textversionierung.

4.5.2 Textversionierung

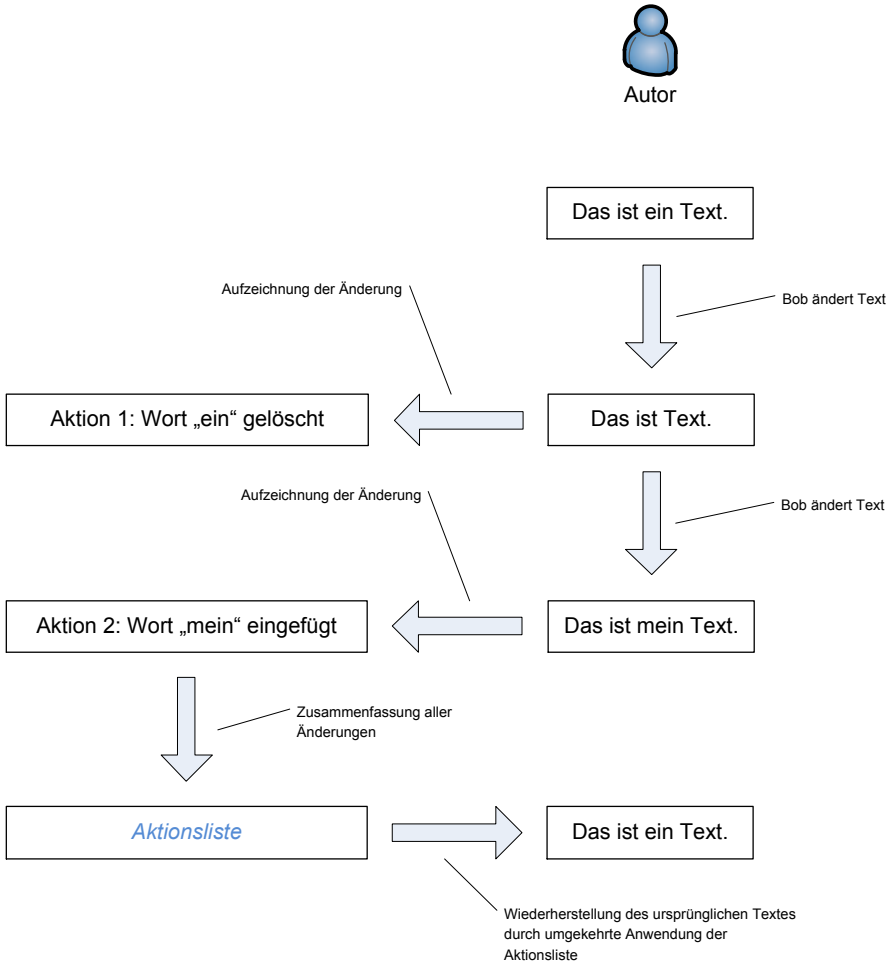

Abbildung 27: Änderungsverfolgung durch Aktionsaufzeichnung

Bei einer Textversionierung wird bei jeder Textsynchronisation eine neue Version des Textes erstellt. Die vorherige Version wird gespeichert und kann bei Bedarf abgerufen werden. Dieses Verfahren ist beispielsweise typisch für Dokumenten-Management-Systeme oder Konfigurationsverwaltungen in der Softwaretechnik. Der Unterscheid zur Aktionsverwaltung liegt darin, dass ausgehend von einer älteren Textversion die getätigten Aktionen dynamisch ermittelt werden können, während bei einer Aktionsaufzeichnung ausgehend vom Aktionslogbuch eine ältere Textversion dynamisch erzeugt werden kann.

Wie die Aktionsaufzeichnung führt auch die Textversionierung zu einem erhöhten Datenaufkommen. Dies kann durch eine Differenzbildung bei der Ablage der Textversionen optimiert werden. Bei einer Differenzbildung wird nicht die Gesamtkopie eines Textes gespeichert, sondern lediglich das Delta zur vorherigen Version. Bei geringfügigen Änderungen in großen Texten kann dies erheblich Speicherplatz sparen.

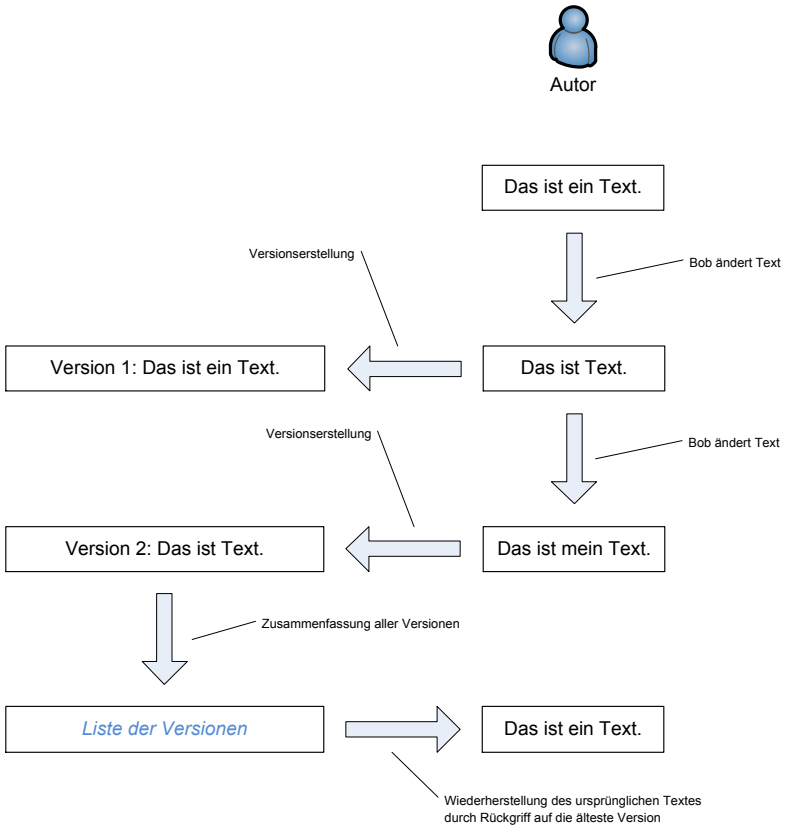

Abbildung 28: Änderungsverfolgung durch Textversionierung

4.6 Schlussbetrachtung

Die Konfliktmanagement-Strategien, die in diesem Kapitel besprochen wurden, lassen sich prinzipiell für jede Systemarchitektur bzw. jede Synchronisationstopologie anwenden. Der Synchronisation oder genauer gesagt der Konfliktauflösung bzw. Konfliktvermeidung kommt dabei eine zentrale Bedeutung innerhalb eines KS-Systems zu. Der Konfliktvermeidung durch Sperren lässt wegen des für andere Autoren gesperrten Textes kein zeitgleiches Bearbeiten desselben zu. Praktikable und umfassende Mechanismen für eine Echtzeitsynchronisation liefern hier nur die OT-Modelle. Das Konfliktauflösen durch Prioritäten bzw. Vereinigung kann ggf. zusätzlich eingesetzt werden, um OT-Algorithmen zu verfeinern. Die automatischen Konfliktauflösungsmechanismen allein können praktisch keine Echtzeitsynchronisation gewährleisten.

Die OT-Modelle basieren auf elementaren Operationen, deren Ausführungseffekt über sogenannte Transformationen so korrigiert wird, dass der korrekte Ausführungseffekt gemäß des ursprünglichen Generierungskontextes erreicht wird. Der OT-Algorithmus entscheidet darüber, welche Transformationen wann ausgeführt werden. OT-Algorithmus, Transformationen und Bedingungen müssen sicherstellen, dass die Texte auf den Clients nicht divergieren. Schon einfache Beispiele zeigen, dass Korrektheit und Konvergenz nicht ganz trivial sind.

Neben Korrektheit und Konvergenz spielt aber auch die Effizienz eines OT-Algorithmus für die Praxis eine wesentliche Rolle. Die Komplexitätsbetrachtungen sind auf den ersten Blick einfach: Im *Worst Case* müssen bei jeder neu anstehenden Transformation die zuvor ausgeführten und im Protokoll vermerkten früheren Transformationen rückgängig gemacht werden. Bei einer Protokolllänge *n* ergibt sich eine Komplexität[11] $O\ n^2$. Auch eine Komplexität $O\ n^2$, die in anderen Anwendungsfeldern durchaus akzeptabel ist, könnte bei einem OT-Modell, das als Datenmodell einen Text als lineare Zeichenkette (*plain text*) versteht und über entsprechend einfache Transformationen verfügt, praktisch durchaus problematisch sein.

Ein weitere Charakteristik der OT-Systeme sind die Bedingungen, die vom OT-Algorithmus eingehalten werden müssen. Die erste Bedingung, Kausalität, geht zurück auf die Zeitordnung in Verteilten Systemen (Betriebssystemen) , genauer gesagt auf die von Leslie Lamport [Lamport 77] beschriebene „happened-before" Relation, hinter der sich nichts anderes als eine partielle Ordnungsrelation darstellt, die eine totale Ordnung aller Ereignisse bzw. Operationen sicherstellt. Es handelt sich also lediglich um eine zeitliche Ordnung, nicht um eine Ursache-Wirkung-Relation wie es der Begriff „Kausalität" anfangs vermuten lässt. Die zweite Bedingung, Konvergenz, ist eigentlich zwingend. Es ist daher nicht ganz nachvollziehbar, dass die CCI-Modelle und CSM-Modelle das Konvergenzkriterium weniger streng auflösen zum Konzept der sogenannten Intuition, die bewusst Divergenz in Kauf nimmt, um formale Vorteile zu gewinnen. Hier wird die Vorteilsgewinnung im mathematischen Formalismus durch den sehr vagen Begriff der Intuition motiviert.

Die OT-Modelle stellen eine Form von Replikation dar wie sie aus Datenbankanwendungen bekannt sind. Bei Datenbanken besteht das Datenmodell aber aus Datensätzen, die in Tabellen angeordnet sind[12]. Die elementaren Operationen bzw. Aktionen *Insert*, *Delete*, *Modify* arbeiten entweder auf Datensätzen oder Feldern eines Datensatzes. Die Benutzer einer Datenbank editieren in der Regel unterschiedliche Datensätze, so dass eine Echtzeit-Synchronisation in der Regel nicht erforderlich ist. Die Synchronisation selbst läuft dann auf Feld bzw. Datensatzebene ab. Bei Gruppen-Editoren steht anstelle der Datenbank das Dokument bzw. der gemeinsam zu bearbeitende Text. Das Datenmodell und die Granularität des Modells ist eine andere: Bei Datenbanken sind es die Tabellen und Datensätze, bei Gruppen-Editoren sind es die Zeichen, Wörter, Paragrafen, je nachdem welches OT-Modell verwendet wird.

Die formalen Methoden der OT-Systeme sind für die Synchronisation von Teiltexten, die von mehreren Autoren verteilt bzw. parallel verändert wurden, sehr nützlich. Je höher der Abstraktionsgrad des Datenmodells, desto effektivere OT-Algorithmen können eingesetzt werden. Es ist daher naheliegend, weitere Strukturinformationen in das Datenmodell mit aufzunehmen, die zum Teil für wissenschaftliche Texte spezifisch sind, z.B. Zusammenfassungen, Kapitel, Quellenverzeichnisse, Zitate, De-

[11] Es wird hier die Landau-O-Notation für Komplexitätsangaben verwendet, die nicht mit der Operatoren-Notation aus den vorhergehenden Kapiteln verwechselt werden sollte.

[12] Mit Datenbank sind im Folgenden relationale Datenbanken gemeint.

finitionen, Beweise, Lemmata usw. Es wäre wahrscheinlich sehr ungeschickt einer Textänderung in einem mathematischen Beweis mit dem Intuitionskonzept zu begegnen, da es hier sehr genau auf Zeichen und die Attribute des Zeichens ankommt (ist es z.B. hochgestellt als Exponent ausgewiesen oder nicht). Bei der Betrachtung der Synchronisationsproblematik sollte hier auch in Erinnerung gerufen werden, dass es neben der methodischen Synchronisation auch eine soziale Synchronisation gibt, die möglichst vom Gruppen-Editor unterstützt werden sollte. So kann sich ein Autor entscheiden, die Schlussbetrachtung allein zu schreiben und weist die anderen Co-Autoren darauf hin, dass vorerst niemand in diesem Teil Änderungen vornimmt. Eine komplizierte Synchronisationsmethodik entfällt: Die Schlussbetrachtung auf dem Server muss lediglich gegen die exklusiv editierte Schlussbetrachtung des Autors ausgetauscht werden.

Die momentan größte Popularität verzeichnen Web-basierte Architekturen wie beispielsweise Wiki-Systeme und Online-Textverarbeitungen. Die Gründe hierfür liegen auf der Hand. Ein Autor benötigt lediglich einen funktionierenden Internetanschluss und einen Web-Browser, um ein solches KS-System zu nutzen. Zudem ist die Bedienung einfach, da die Funktionalität technisch bedingt sehr übersichtlich ist und die Bedienung sich nicht wesentlich von anderen Anwendungen oder Diensten im Web unterscheidet. Außerdem erlaubt das direkte Arbeiten im Web-Browser die problemlose Nutzung paralleler Dienste wie z.B. Chat. Die mit OT-Systemen diskutierte Methodik steht erst seit kurzer Zeit in professionellen Anwendungen wie Microsoft Word 2010 und Google Text der wissenschaftlichen wie der nicht-wissenschaftlichen Gemeinschaft zur Verfügung.

5 KS-Software-Architekturen

5.1 Motivation

Nach dem in Kapitel 4 die Entwurfsmuster diskutiert wurden, welche die Bausteine darstellen, aus denen ein KS-System besteht, wird in diesem Kapitel auf die konkrete Architektur eines KS-Systems eingegangen.

Die Ursprünge heutiger Textverarbeitungssysteme wie Microsoft Word oder Open Office reichen zurück in die Zeit, als das Internet noch bedeutungslos war bzw. die Übertragungsraten von Rechnernetzwerken nur einem Bruchteil heutiger Kapazitäten entsprachen. Software-Anwendungen wurden zu dieser Zeit für den Einzelanwender konzipiert, der seine Daten per externen Datenträger von einem Rechner auf den anderen Rechner übertragen hat. Aus dieser Ein-Benutzer-zentrierten Sichtweise haben sich die meisten Systeme nie wirklich gelöst, trotz innovativer Weiterentwicklung in vielen anderen Bereichen (z.B. Gestaltung der Benutzeroberflächen).

Aktuell steht man als Autor also dem folgenden Dilemma gegenüber: Auf der einen Seite wächst der Drang nach Kooperation mit anderen Autoren, auf der anderen Seite ist das Erstellen umfangreicher Textpublikationen auf Softwaresysteme mit eingeschränkten bzw. nicht vorhandenen kollaborativen Fähigkeiten beschränkt. Das Ergebnis dieses Dilemmas ist in der Regel ein Zurückgreifen auf sub-optimale Lösungen:

- Das Beispiel in Kapitel 2.2.1 beschreibt die Nutzung eines artikel-basierten Wiki-Systems für das kollaborative Schreiben einer umfangreichen wissenschaftlichen Publikation.

- Das Beispiel in Kapitel 2.3.1 zeigt, wie der Versuch unternommen wurde, das Fehlen von Kollaborationswerkzeugen beim Erstellen eines Ausschreibungstextes durch Nutzung von E-Mail und zusätzlicher Man-Power auszugleichen.

Obwohl wir uns in einem Zeitalter des *Ubiquitous Computing*[13] befinden, der Computer also allgegenwärtig ist und das Internet uns zeitlich uneingeschränkte Kommunikationsfähigkeiten ermöglicht, so ist es beispielsweise heute immer noch äußerst schwierig, ein Buchprojekt durch reaktive Texterstellung zu realisieren. Dabei liegt das Problem nicht in der Technik:

[13] Der Ursprung dieses Begriffs lässt sich auf den 1991 erschienen Artikel „The Computer for the 21st Century“ [Weiser 91] zurückverfolgen.

- Die in Kapitel 4 ausgearbeiteten Entwurfsmuster zeigen, dass es für nahezu alle Anforderungen (im engeren Sinne) an ein KS-System passende Software-Lösungen gibt.

- Nahezu alle Anforderungen (im weiteren Sinne) an ein KS-System sind bereits in existierenden Software-Lösungen umgesetzt, die lediglich geschickt miteinander kombiniert werden müssen.

Die Gründe müssen also woanders gesucht werden. Sie scheinen mir unter anderem in einer fehlenden Standardisierung von Textverarbeitungssystemen zu liegen. Ein Beispiel soll dies verdeutlichen: Eine der populärsten Anwendungen im Internet ist E-Mail. E-Mail baut sich um die weltweit akzeptierten Protokollstandards POP3 (RFC 1939), IMAP4 (RFC 3501) und SMTP (RFC 5321) herum auf. Dabei ist es egal, welchen konkreten E-Mail-Client man verwenden möchte, es ist immer sichergestellt, dass die eigene E-Mail den Adressaten erreicht, solange die Kommunikation die Protokollspezifikationen einhält. E-Mail ist, obwohl schon relativ alt, nach wie vor das wohl am häufigsten genutzte Kommunikationswerkzeug im Internet.

Ein Blick auf die Textverarbeitung zeigt ein anderes Bild. Professionelle Textverarbeitungssysteme sind in der Regel geschlossene Systeme, die nicht beliebig untereinander ausgetauscht werden können. Das beginnt schon damit, dass die Dateiformate trotz funktionaler Übereinstimmungen in vielen Bereichen syntaktisch inkompatibel zueinander sind. Zwar existieren mittlerweile standardisierte Dokumentformate wie ODF [ODF 09] oder Office Open XML [ECMA 08], diese haben sich aber immer noch nicht flächendeckend durchgesetzt, so dass sie weiterhin an ihr ursprüngliches Textverarbeitungssystem gebunden sind. Zudem umfassen diese Spezifikationen nur sehr eingeschränkt kollaborative Erweiterungen. Über das Warum lässt sich nur spekulieren, allerdings dürfte die Marktbeherrschung von Microsoft als Hersteller von Microsoft Word einen nicht unerheblichen Anteil daran haben. Das Interesse an einer größeren Interoperabilität von Textverarbeitungssystemen dürfte von dieser Seite her nicht sehr ausgeprägt sein.

Im Folgenden werden daher zwei alternative Software-Architekturen beschrieben, die als weiterentwickelte Vorschläge für die Entwicklung verbesserter KS-Systeme zu verstehen sind. Beide Ansätze unterscheiden sich grundsätzlich voneinander:

- Der erste Ansatz besteht in der Erweiterung des bereits etablierten Protokollstandards WebDAV zum Austausch von Dokumenten. Die Erweiterung namens TextDAV besteht darin, die Kooperationseinheit „Textdokument“ um eine weitere Kooperationseinheit „Textabschnitt“ zu ergänzen. Dies erlaubt das Synchronisieren nicht nur von ganzen Textdokumenten sondern auch von beliebigen Abschnitten innerhalb eines Textdokuments.

- Der zweite Ansatz namens C-LAB ist abstrakter und verlässt die fokussierte Spezifikationsebene eines Protokolls. Er skizziert das Design eines vollständig integrierten kollaborativen Textverarbeitungssystems unter Berücksichtigung der in Kapitel 3.3 herausgearbeiteten Anforderungen. Eine Referen-

zimplementierung erfolgt in diesem Rahmen allerdings nicht, da sie den Rahmen der vorliegenden Arbeit sprengen würde.

5.2 Text Extensions for WebDAV (TextDAV)

5.2.1 Einführung

TextDAV ist eine Erweiterung des im RFC 4918 [RFC 4918] definierten WebDAV-Protokollstandards. WebDAV erweitert das im RFC 2616 [RFC 2616] beschriebene Kommunikationsprotokoll HTTP um die Möglichkeit, Dokumente über das Internet konfliktfrei zu organisieren bzw. zu bearbeiten. WebDAV wird vor allem im Kontext von Dokumenten-Management-Systemen als standardisierte Kommunikationsschnittstelle eingesetzt, kann aber auch lediglich als Erweiterung eines herkömmlichen Web-Servers (z.B. Internet Information Server oder Apache) implementiert werden. Zunächst soll jedoch das HTTP-Protokoll (*Hypertext Transfer Protocol*) näher betrachtet werden.

Unter HTTP werden alle Objekte als Ressourcen betrachtet, auf die per URL (*Uniform Resource Identifier*) zugegriffen werden kann. Operationen mit diesen Objekten werden ausgeführt, indem HTTP-Anfragen (*HTTP-Requests*) an den HTTP-Server gesendet werden, welche die angeforderte Methode ausführen. Der HTTP-Server beantwortet die Anfrage mit einer HTTP-Antwort (*HTTP-Response*). Anfragen und Antworten müssen einer vorgegebenen Syntax entsprechen.

```
GET /projects/dissertation.xml http/1.1
Host: www.myserver.de
↵
```

Beispiel 5-1: Beispiel für einen HTTP-GET-Request

```
HTTP/1.0 200 OK
Date: Fri, 5 Dec 2010 08:00:11 GMT
Last-Modified: Tue, 2 Dec 2010 11:11:11 GMT
Content-Language: de
Content-Type: text/html; charset=utf-8
↵
Mein Text
```

Beispiel 5-2: Beispiel für eine HTTP-GET-Response

Für das kollaborative Arbeiten sind lediglich die folgenden HTTP Methoden sinnvoll:

- GET – Fordert eine durch die als Parameter angegebene URL spezifizierte Ressource vom Server an.

- PUT – Erzeugt eine neue Ressource an dem durch die URL angegebenen Ort bzw. aktualisiert eine Ressource an diesem Ort. Oftmals wird dieser Befehl aber vom Server-Administrator gesperrt, da alternativ der POST-Befehl besser geeignet ist.

- DELETE – Löscht eine Ressource an der angegebenen URL, wird aber wie auch der PUT-Befehl oftmals vom Server-Administrator gesperrt, da seine Funktionalität auch über POST erreicht werden kann.

- POST – Dieser Befehl ist der allgemeinste der hier vorgestellten. Er enthält eine URL als Parameter und spezifiziert damit den Aufruf einer Ressource oder eines Skriptes. Beim Aufruf können vom Prinzip beliebige weitere Parameter an das Skript übergeben werden. Über den Aufruf eines Skriptes kann beliebige Funktionalität angefordert werden, somit auch die, welche alternativ durch PUT und DELETE erreicht werden kann.

- HEAD – Dieser Befehl liefert die Header-Informationen der Antwort über die per URL angegebene Ressource, ohne diese selbst anzufordern.

- OPTIONS – Mit einer OPTION-Request ist es dem Client möglich vom Server zu erfahren, welche Befehle denn überhaupt möglich sind, also z.B. GET, HEAD, OPTIONS.

Eine vollständige Beschreibung aller HTTP–Befehle kann dem Standard (siehe *[RFC 1945], [RFC 2616]*) entnommen werden.

Für die Synchronisation kommt in HTTP dem sogenannten ETag (entity tag) eine entscheidende Bedeutung zu. Es handelt sich dabei um eine eindeutige Identifikation für eine Ressource, die vom Server bei jeder Änderung an der Ressource neu zugewiesen wird. Über das ETag kann die Synchronisation von gleichzeitig bearbeiteten Ressourcen bzw. Dokumenten erreicht werden, indem der Server eine PUT-Anfrage des Clients für das Dokument ablehnt, wenn sich das ETag des Dokuments auf dem Server von dem ETag des Client-Dokuments unterscheidet. In diesem Fall ist das Dokument auf dem Server nämlich bereits von einem anderen Autor geändert worden und die vorliegende GET-Request wird abgelehnt, um Konflikte zu vermeiden. Dieses Vorgehen entspricht dem optimistischen Sperren (siehe hierfür auch Kapitel *4.4.5*).

WebDAV erweitert nun das HTTP-Datenmodell um die Elemente *Collections*, *Properties* und *Locks*:

- **Collection** – Eine Collection ist eine hierarchische Sammlung von Ressourcen, die ihrerseits wieder Ressourcen enthalten können. Eine Collection ist über ihre URL adressierbar. Eine Collection kann analog zu einem Verzeichnisbaum unter UNIX oder Windows gesehen werden.

- **Properties** – Alle Ressourcen haben Eigenschaften (*properties*) bestehend aus *Name* und *Wert*. Diese Eigenschaften sind allerdings nicht adressierbar und haben keine URL. Eigenschaften repräsentieren Metadaten, d.h. Daten über Daten, zu einer Ressource, z.B. ein beschreibendes Schlagwort zu einer Bilddatei. WebDAV unterscheidet dabei „*Live*“- und „*Dead*“- Eigenschaften:

- **Live Property –** Eine Eigenschaft, deren Syntax und Semantik vom Server unterstützt wird, z.B. die „getcontentlength“ Property, welche die Größe eines Inhalts aufgrund eines HTTP GET-Befehls zurückliefert. In WebDAV müssen je Ressource die folgenden Live-Properties zwingend vorhanden sein (die Bezeichner lassen die Bedeutung vermuten, eine genaue Beschreibung findet sich in RFC 4918 [RFC 4918]):

  creationdate
  displayname
  getcontentlanguage
  getcontentlength
  getcontenttype
  getetag
  getlastmodified
  resourcetype
  lockdiscovery
  source
  supportedlock

- ***Dead Property –*** Eine Eigenschaft, deren Syntax und Semantik vom Server nicht unterstützt wird. Der Server vermerkt nur den Wert der Eigenschaft und nur der Client kennt die Syntax und Semantik und kann die Eigenschaft verarbeiten.

- **Locks** – Jede Ressource kann gegenüber Änderungen durch andere Benutzer per Lock gesperrt werden. Locks selbst sind ebenfalls nicht adressierbar und besitzen keine URL. In WebDAV werfen nur Schreib-Locks definiert. In der Regel erlaubt der Server nur den Benutzern Locks zu setzen, die auch dafür die entsprechenden Rechte haben.

Die Erweiterung des HTTP-Datenmodells durch WebDAV geht einher mit einer Erweiterung der Semantik und der Funktionalität:

- Neue Status und Fehlermeldungen als Response auf eine http-Methode
- Die neue Methode MKCOL, die es erlaubt, eine neue Collection zu erzeugen.
- Zwei neue Methoden PROPFIND und PROPMATCH erlauben es, in einer Collection-Hierarchie eine Eigenschaft zu finden bzw. zu verändern.
- Detailliertere Möglichkeiten einen Lock auf einer Collection zu erzeugen.
- Die Möglichkeit genau anzugeben, welche Eigenschaftswerte zurückgegeben werden, damit das Listing in der Antwort nicht zu allgemein und damit zu unübersichtlich wird.
- Funktionen für diverse bereits existierende Web-Authoring-Tools, damit diese WebDAV integrieren können

Zusammengefasst erleichtern diese Erweiterungen von HTTP das Organisieren und konfliktfreie Synchronisieren von Dateien bzw. Dokumenten in einer kollaborativen Arbeitsumgebung.

.

TextDAV erweitert nun die Granularität von WebDAV, indem das jeweilige Textdokument bzw. die jeweilige Ressource nicht als Ganzes betrachtet wird, sondern die kleineste Einheit in dem Modell eine beliebige Untermenge des Textes bzw. ein Abschnitt des Textes ist. Genauer gesagt erweitert TextDAV den WebDAV-Standard, indem das Datenmodell und die Methoden erweitert werden:

1. TextDAV unterscheidet zwischen speziellen TextDAV-Dokumenten und normalen WebDAV-Ressourcen. Während der Inhalt von WebDAV- Ressourcen beliebig sein kann, enthalten TextDAV-Dokumente XML-formatierte Texte, die nach den Richtlinien der *Text Encoding Initiative* (TEI) ausgezeichnet sind. Analog dazu, dass WebDAV den HTTP-Standard als echte Teilmenge subsummiert, ist TextDAV eine echte Obermenge von WebDAV.

2. TextDAV kann beliebige Textabschnitte innerhalb eines TextDAV-Dokumentes an Hand von speziellen XML-Attributen erkennen. Ein Textabschnitt kann wiederum andere Textabschnitte enthalten und somit die Hierarchie eines Textes bestehend aus Kapiteln und Unterkapiteln repräsentieren. Diese Textabschnitte bzw. Teilbäume einer Textabschnittshierarchie können ähnlich wie normale WebDAV-Dokumente zentral verwaltet und gesperrt werden.

3. Zum gezielten Verwalten dieser Teiltexte definiert TextDAV zusätzliche Methoden, welche den vorhandenen WebDAV-Befehlssatz erweitern.

4. Jeder Teiltext ist einem Eigentümer zugeordnet und wird über eine ID eindeutig identifizierbar.

5. Schreibrechte werden implizit mit Hilfe der Hierarchie der Textabschnitte und einer eindeutigen Zuordnung eines jeden Textabschnitts zu seinem Eigentümer erteilt.

5.2.2 Begriffsdefinition

Die folgenden Begriffe sind von zentraler Bedeutung und sind wie folgt zu verstehen:

- **TextDAV-Client** – Ein WebDAV-Client (z.B. ein Textverarbeitungssystem), der zusätzlich die Erweiterungen der TextDAV-Spezifikation implementiert.

- **TextDAV-Server** – Ein WebDAV-Server (z.B. ein Web-Server), der zusätzlich die Erweiterungen der TextDAV-Spezifikation implementiert.

- **TextDAV-Dokument** – Eine Web-DAV-Ressource, dessen Inhalt nach den Richtlinien der *Text Encoding Initiative* (TEI) ausgezeichnet wurde. Ein Text-DAV-Dokument ist ein XML-Dokument.

- **TextDAV-Teiltext** – Ein durch definierte Auszeichnungen (XML-Markups) kenntlich gemachter markierter Teilbereich innerhalb eines TextDAV-Dokuments.

- **TextDAV-Eigentümer** – Ein Benutzer, welcher das ausschließliche Schreibrecht für einen TextDAV-Teiltext besitzt. TextDAV-Eigentümer können einen Wechsel der Eigentum-Zuordnung initiieren.

- **TextDAV-Befehle** – Drei HTTP-Methoden, die den Befehlssatz der WebDAV-Spezifikation ergänzen.

5.2.3 TextDAV-Dokumente

Ein TextDAV-Dokument ist ein XML-Dokument, das nach den Richtlinien der *Text Encoding Initiative* (TEI) ausgezeichnet ist [TEI 07]. TEI ist ein im Jahre 1987 gegründetes Konsortium, dessen Ziel die Definition von einheitlichen Textauszeichnungen in der Geisteswissenschaft ist. Die Liste der Mitglieder ist lang und umfasst vor allem Universitäten und Bibliothekseinrichtungen. TEI ist auch gleichzeitig der Name eines Richtlinienkatalogs, der ein umfassendes Auszeichnungsschema für verschiedenste Typen von Texten definiert. Dieser Richtlinienkatalog liegt im Moment in der Version P5 (vom November 2007) vor.

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>TextDAV-Beispiel</title>
   </titleStmt>
   <publicationStmt>
    <p>Minimales Beispiel für ein TextDAV-Dokument</p>
   </publicationStmt>
  </fileDesc>
 </teiHeader>
 <text>
  <body>
   <p>Das ist ein Beispiel für ein TextDAV-Dokument.</p>
  </body>
 </text>
</TEI>
```

Beispiel 5-3: Minimales Text-DAV-Dokument

Der Text besteht lediglich aus einem beschreibenden Header (<teiHeader>… </teiHeader>) und dem Textkörper (<text>...</text>). Eine ausführliche Erläuterung aller XML-Elemente von TEI findet sich in den TEI-Richtlinien [TEI 07]. Das Auszeichnungsschema von TEI lässt sich mit den gängigen XML-Metabeschreibungen DTD, XML-Schema und RELAX NG ausdrücken.

5.2.4 TextDAV-Abschnitte

TEI erlaubt es, ein Textdokument in logische Abschnitte (divisions) zu unterteilen und diese hierarchisch zu gliedern. Es wird zwischen nummerierten Abschnitten (numbered divisions) und nicht-nummerierten Abschnitten (unnumbered divisions) unterschieden. TextDAV-Teiltexte sind nicht-nummerierte Abschnitte, welche der folgenden Syntax entsprechen müssen:

```
<div type="textdav-part" id=1234 owner="Frank">
```

Das -Auszeichnungselement weist demnach drei Besonderheiten auf:

1. Die Eigenschaft „type“ muss zwingend den Wert „textdav-part“ aufweisen. Nur so kann TextDAV zwischen Abschnitten, die einen Teiltext auszeichnen, und Abschnitten, die keinen Teiltext auszeichnen unterscheiden.

2. Die Eigenschaft „id“ muss einen eindeutigen Wert besitzen, der es erlaubt einen Teiltext von allen anderen Teiltexten innerhalb eines TextDAV-Dokumentes eindeutig zu unterscheiden.

3. Das nicht in TEI definierte Attribut „owner“ ist optional und enthält als Wert den Namen des Eigentümers eines TextDAV-Teiltextes. Es gilt die Regel, dass die Eigenschaft „owner“ nur dann enthalten ist, wenn der TextDAV-Client keinen Schreibzugriff auf dieses Textsegment hat. In diesem Fall verrät die Eigenschaft „owner“, wer die Erlaubnis zum Schreiben hat.

Das folgende Beispiel zeigt ein TextDAV-Dokument, dass aus zwei Teiltexten besteht, deren Eigentümer jeweils „Markus“ und „Lara“ sind:

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>TextDAV-Beispiel</title>
   </titleStmt>
   <publicationStmt>
    <p>Minimales Beispiel für ein TextDAV-Dokument</p>
   </publicationStmt>
  </fileDesc>
 </teiHeader>
 <text>
  <body>
   <div type="textdav-part" id=1234 owner="Markus">
     <head>Kapitel 1</head>
     <p>Text für Kapitel 1</p>
   </div>
   <div type="textdav-part" id=1235 owner="Lara">
     <head>Kapitel 2</head>
     <p>Text für Kapitel 2</p>
   </div>
  </body>
 </text>
</TEI>
```

Beispiel 5-4: Erweitertes Beispiel mit zwei Eigentümern „Markus“ und „Lara“

Teiltexte können auch ineinander verschachtelt werden. Dies erlaubt eine hierarchische Unterteilung des Gesamttextes und eine implizite Rechteverteilung. Dabei gelten folgende Regeln:

- Alle TextDAV-Teiltexte können von allen Benutzern, die Zugriff auf das TextDAV-Dokument haben, gelesen werden.
- Ein TextDAV-Teiltext kann nur von seinem Eigentümer geändert werden.
- Enthält ein TextDAV-Teiltext weitere Teiltexte mit abweichenden Eigentümern, so kann der Eigentümer des übergeordneten Teiltextes diese löschen aber nicht verändern.
- Der Wechsel eines TextDAV-Eigentümers kann nur von seinem aktuellen Eigentümer initiiert werden.

Das folgende Beispiel zeigt ein TextDAV-Dokument mit verschachtelten Teiltexten und unterschiedlichen Eigentümern:

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>TextDAV-Beispiel</title>
   </titleStmt>
   <publicationStmt>
    <p>Minimales Beispiel für ein TextDAV-Dokument</p>
   </publicationStmt>
  </fileDesc>
 </teiHeader>
 <text>
  <body>
   <div type="textdav-part" id=1000 owner="Markus">
     <head>Kapitel 1</head>
     <p>Text für ersten Absatz</p>
     <p>Text für zweiten Absatz</p>
     <div type="textdav-part" id=1010 owner="Lara">
      <p>Text für dritten Absatz</p>
     </div>
   </div>
   <div type="textdav-part" id=2000 owner="Lara">
     <head>Kapitel 2</head>
     <p>Text für Kapitel 2</p>
   </div>
  </body>
 </text>
</TEI>
```

Beispiel 5-5: Verschachtelte Teiltexte mit unterschiedlichen Eigentümern

In diesem Beispiel kann der Eigentümer „Markus“ von Teiltext 1000 nur die ersten beiden Absätze bearbeiten, nicht aber den Teiltext 1010. Umgekehrt kann die Eigentümerin „Lara“ des Teiltextes 1010 nicht den Text der höheren Gliederungsebene bearbeiten. Beide Zugriffsbereiche schließen sich also gegenseitig aus. Eine derartige Verschachtelung kann auch über mehr als eine Ebene definiert werden.

5.2.5 TextDAV-Requests

Da ein TextDAV-Dokument gleichzeitig ein normales WebDAV-Dokument ist, kann dieses mit den Standardbefehlen von WebDAV erstellt, geöffnet, geschlossen, gelöscht, kopiert und umbenannt werden. Für das Laden und Speichern von TextDAV-Teiltexten sowie den Wechsel des Eigentümers definiert TextDAV jedoch zusätzliche HTTP-Methoden:

- **GETPART:** Lädt ein oder mehrere TextDAV-Teiltexte von einem TextDAV-Server herunter. Korrespondiert logisch gesehen mit der WebDAV-Methode GET.
- **PUTPART:** Lädt ein oder mehrere TextDAV-Teiltexte auf einen TextDAV-Server hoch. Korrespondiert logisch gesehen mit der WebDAV-Methode PUT.
- **CHANGEPARTOWNER:** Initiiert den Wechsel des Eigentümers für einen oder mehrere TextDAV-Teiltexte.

Alle drei Methoden und ihre jeweils nötigen Aufrufparameter werden im Folgenden beschrieben.

5.2.5.1 GETPART

GETPART ermöglicht einem TextDAV-Client das Herunterladen von TextDAV-Teiltexten aus einem TextDAV-Dokument.

Das folgende Beispiel lädt ein Textsegment von einem TextDAV-Server:

```
GETPART /projects/dissertation.xml http/1.1
Host: www.myserver.de
Content-Type: text/xml
Content-Length: xxx
↵
<?xml version="1.0" encoding="utf-8">
<parts xmlns='DAV:'>
 <part id='1234'/>
</parts>
```

Es wird der TextDAV-Teiltext mit der ID 1234 aus dem TextDAV-Dokument /projects/dissertation.xml auf dem WebDAV-Server www.myserver.com geladen.

Sollen mehrere TextDAV-Teiltexte mit einem Befehl heruntergeladen werden, kann die Liste im XML-Payload um weitere Einträge erweitert werden. Im folgenden Beispiel werden drei TextDAV-Teiltexte auf einmal angefragt.

```
GETPART /projects/dissertation.xml http/1.1
Host: www.myserver.de
Content-Type: text/xml
Content-Length: xxx
↵
<?xml version="1.0" encoding="utf-8">
<parts xmlns='DAV:'>
 <part id='1234'/>
 <part id='1235'/>
 <part id='1238'/>
</parts>
```

Eine erfolgreiche GETPART-Antwort erhält der TextDAV-Client, wenn der TextDAV-Server alle übergebenen TextDAV-Teiltexte ausfindig machen konnte und sie somit zurückliefern kann. TextDAV-Teiltexte werden gefunden, wenn

- die WebDAV-Ressource auf dem TextDAV-Server existiert und der aktuelle Benutzer Leserechte für diese Ressource besitzt.
- der Inhalt der WebDAV-Ressource eine TEI-Formatierung aufweist und somit als TextDAV-Dokument interpretiert werden kann.
- das TextDAV-Dokument alle referenzierten TextDAV-Teiltexte enthält.

Eine mögliche positive Antwort könnte wie folgt aussehen:

```
HTTP/1.0 207 MultiPart Response
Date: Sun, 29 Jun 2009, 12:00:00 GMT
Content-Type: text/xml; charset=UTF-8
Content-Length: xxx
↵
<?xml version="1.0" encoding="utf-8">
<parts xmlns='DAV:'>
 <part id='123456'>
  Das ist der Text einer Textsequenz.
 </part>
 <part id='123457'>
  Das ist ebenfalls Text einer Textsequenz.
 </part>
 <part id='123459'>
  Das ist nochmals Text einer Textsequenz.
 </part>
</parts>
```

5.2.5.2 PUTPART

PUTPART ermöglicht einem TextDAV-Client das Speichern von TextDAV-Teiltexten in einem TextDAV-Dokument auf einem TextDAV-Server.

Das folgende Beispiel speichert ein Textsegment auf einem TextDAV-Server:

```
PUTPART /projects/dissertation.xml http/1.1
Host: www.myserver.de
Content-Type: text/xml
Content-Length: xxx
```

```
↵
<?xml version=”1.0” encoding=”utf-8”>
<parts xmlns=’DAV:’>
 <part id=’123456’>
  Das ist der Text einer Textsequenz.
 </part>
</parts>
```

Es wird der TextDAV-Teiltext mit der ID 1234 an den TextDAV-Server unter www.myserver.com gesendet, der ihn wiederum im TextDAV-Dokument /projects/dissertation.xml speichern soll.

Sollen mehrere TextDAV-Teiltexte mit einem Befehl gespeichert werden, kann die Liste im XML-Payload um weitere Einträge erweitert werden. Im folgenden Beispiel werden drei TextDAV-Teiltexte auf einmal gespeichert.

```
PUTPART /projects/dissertation.xml http/1.1
Host: www.myserver.de
Content-Type: text/xml
Content-Length: xxx
↵
<?xml version=”1.0” encoding=”utf-8”>
<parts xmlns=’DAV:’>
 <part id=’123456’>
  Das ist der Text einer Textsequenz.
 </part>
 <part id=’123457’>
  Das ist ebenfalls Text einer Textsequenz.
 </part>
 <part id=’123459’>
  Das ist nochmals Text einer Textsequenz.
 </part>
</parts>
```

Eine erfolgreiche PUTPART-Antwort erhält der TextDAV-Client, wenn der TextDAV-Server alle übergebenen TextDAV-Teiltexte erfolgreich speichern konnte. TextDAV-Teiltexte werden erfolgreich gespeichert, wenn

- die WebDAV-Ressource auf dem TextDAV-Server existiert und der aktuelle Benutzer Schreibrechte für diese Ressource besitzt.
- der Inhalt der WebDAV-Ressource eine TEI-Formatierung aufweist und somit als TextDAV-Dokument interpretiert werden kann.
- das TextDAV-Dokument die referenzierten TextDAV-Teiltexte auch wirklich enthält.
- die momentanen Eigentümer der TextDAV-Teiltexte identisch sind mit dem aktuellen Benutzer des TextDAV-Clients.

Eine mögliche positive Antwort könnte wie folgt aussehen:

```
HTTP/1.0 204 No Content
Date: Sun, 29 Jun 2009, 12:00:00 GMT
Content-Length: 0
```

5.2.5.3 *CHANGEPARTOWNER*

CHANGEPARTOWNER ermöglicht einem TextDAV-Client den Eigentümer von TextDAV-Teiltexten in einem TextDAV-Dokument auf einem TextDAV-Server zu ändern. Durch den Wechsel des Eigentümers, verliert der alte Eigentümer das Recht, diesen Teiltext weiterhin zu ändern.

Das folgende Beispiel ändert den Eigentümer eines TextDAV-Teiltexts auf einem TextDAV-Server:

```
CHANGEPARTOWNER /projects/dissertation.xml http/1.1
Host: www.myserver.de
Content-Type: text/xml
Content-Length: xxx
↵
<?xml version="1.0" encoding="utf-8">
<parts xmlns='DAV:'>
 <part id='1234'newowner="Lida"/>
</parts>
```

Der TextDAV-Teiltext mit der ID 1234 im TextDAV-Dokument /projects/dissertation.xml auf dem TextDAV-Server unter www.myserver.com bekommt einen neuen Eigentümer. Sollen mehrere TextDAV-Teiltexte mit einem Befehl gesperrt werden, kann die Liste (analog zum Beispiel für GETPART) im XML-Payload um weitere Einträge erweitert werden.

Eine erfolgreiche CHANGEPARTOWNER-Antwort erhält der TextDAV-Client, wenn der TextDAV-Server für alle übergebenen TextDAV-Teiltexte den Eigentümer erfolgreich ändern konnte. Teilnehmer für TextDAV-Teiltexte werden erfolgreich geändert, wenn

- die WebDAV-Ressource auf dem TextDAV-Server existiert und nicht für Veränderungen gesperrt ist.
- der Inhalt der WebDAV-Ressource eine TEI-Formatierung aufweist und somit als TextDAV-Dokument interpretiert werden kann.
- das TextDAV-Dokument die referenzierten TextDAV-Teiltexte auch wirklich enthält.
- die momentanen Eigentümer der TextDAV-Teiltexte identisch sind mit dem aktuellen Benutzer des TextDAV-Clients.
- die neuen Eigentümer auf dem TextDAV-Server existieren.

Eine mögliche positive Antwort könnte wie folgt aussehen:

```
HTTP/1.0 204 No Content
Date: Sun, 29 Jun 2009, 12:00:00 GMT
Content-Length: 0
```

5.2.6 TextDAV-Zugriffskontrolle

Die Zugriffkontrolle wird – wie im vorangegangenen Kapitel beschrieben – über die implizite Rechteverteilung, basierend auf der hierarchischen Textstruktur, realisiert: Ein Teiltext kann nur von seinem Eigentümer geändert werden. Ein übergeordneter Eigentümer kann nur löschen aber nicht verändern.

5.2.7 Diskussion

TextDAV erweitert den WebDAV-Standard indem die Textstruktur einer Ressource mit in das Datenmodell einbezogen wird und entsprechende Methoden dazu neu definiert werden. TextDAV als Erweiterung von WebDAV macht keine Aussage darüber, wie ein TextDAV-Server oder ein TextDAV-Client zu implementieren sind. Es wird lediglich ein Kommunikationsprotokoll und eine gemeinsame Textcodierung definiert. Gleichzeitig lassen sich jedoch einige zwingende technische Rahmenbedingungen ableiten.

Während HTTP und WebDAV die feinste granulare Einheit die Datei bzw. das Dokument darstellt, kann ein WebDAV Protokoll ein bereits bestehendes Textverarbeitungssystem zu einem KS-System erweitern, ohne dass die bestehende Anwendung verändert werden muss: WebDAV arbeitet auf den vom Textverarbeitungssystem erzeugten Dateien. Demgegenüber muss ein TextDAV-Client in der Lage sein, TEI-konforme XML-Dokumente zu verarbeiten. Dies ist ein fundamentaler Unterscheid zu normalen WebDAV-Servern, die zwar über die Properties Metadaten über den Inhalt enthalten, die aber zu keinem Zeitpunkt den Inhalt einer WebDAV-Ressource verarbeiten müssen. Ein bestehendes Textverarbeitungssystem müsste also um die Fähigkeit erweitert werden WebDAV-konforme Dokumente zu erzeugen. Dies ließe sich vermeiden, indem der Text lokal weiterhin in seinem originären Format auf dem TextDAV-Client gespeichert wird und lediglich während der Kommunikation mit einem TextDAV-Server als TextDAV-Dokument formatiert wird. Dies setzt allerdings voraus, dass eine 1:1-Abbildung zwischen TextDAV-Dokument und originärem Format existiert.

TextDAV-Dokumente besitzen keine Layout-Informationen. Damit unterscheiden sich TextDAV-Dokumente von gängigen Standards wie OpenDocument [ODF 09] oder Office Open XML [ECMA 08], welche Text und Layout zusammen speichern. Durch die explizite Auszeichnung aller Textelemente in einem TextDAV-Dokument können diese jederzeit in das gewünschtes Druck- oder Anzeigeformat überführt werden. Dabei würde es sich anbieten, die Layout-Informationen als separate WebDAV-Ressourcen zu speichern.

Naheliegender ist es, das Textverarbeitungssystem um entsprechende TextDAV-Funktionen zu erweitern. Dies führt in logischer Konsequenz dazu, dass ein Textverarbeitungssystem als integraler Bestandteil einer KS-System-Architektur be-

trachtet werden sollte. Die im nachfolgenden Kapitel vorgestellte Softwarearchitektur verfolgt genau diesen Aspekt weiter.

5.3 Software Architecture for Collaborative Writing (C-LAB)

5.3.1 Einführung

C-LAB ist eine Software-Architektur, die einen ganzheitlichen Blick auf kollaborative Textverarbeitung wirft. Im Mittelpunkt stehen nicht das Textdokument und seine Ausprägungen, sondern die effektive Organisation eines KS-Projektes unter allen Beteiligten (z.B. Autoren oder Projektleiter) und deren Bedürfnissen (siehe hierzu Kapitel 3.3.2).

C-LAB unternimmt nicht den Versuch, die klassische Textverarbeitung neu zu erfinden, sondern will vielmehr diese in einen größeren kollaborativen Rahmen einbinden. C-LAB entwirft dafür ein Modell von elementaren Bausteinen, die in Beziehung zueinander stehen. Diese Bausteine stellen die meiner Meinung nach essentiellen Grundpfeiler einer idealtypischen kollaborativen Textverarbeitung da.

Zunächst soll jedoch die Frage beantwortet werden, was eine idealtypische kollaborativen Textverarbeitung ausmacht. Dazu wird auf die Klassifikation von Anforderungen aus Kapitel 3.3 zurückgegriffen. Demnach sollte eine kollaborativen Textverarbeitung folgende Aspekte möglichst gut abbilden:

1. Organisation eines KS-Projektes
2. Kommunikation innerhalb eines KS-Projektes
3. Synchronisation unter allen Beteiligten eines KS-Projektes
4. Nachvollziehbarkeit von Aktivitäten innerhalb eines KS-Projektes
5. Schutz gegen unberechtigten Zugriff von außen wie von innen

C-LAB greift sich nun Punkt 1, also die Organisation von KS-Projekten, heraus, da dies der Ausgangspunkt für die Realisierung aller weiteren Punkte ist. Im Folgenden werden nach einer notwendigen Begriffsdefinition die Bausteine einer C-LAB-kompatiblen Textverarbeitung genauer betrachtet. Das Beziehungsgeflecht der Bausteine untereinander wird in Form von einfachen UML-Klassen-Diagrammen visualisiert.

5.3.2 Begriffsdefinition

Für die folgenden Betrachtungen werden folgende Begriffe eingeführt:

- ***C-LAB-Benutzer:*** Ein C-LAB-Benutzer ist ein Akteur innerhalb eines C-LAB-Systems. Durch das Zuweisen von Berechtigungen kann jeder Benutzer eine

bestimmte Rolle innerhalb eines Projektes einnehmen (z.B. Projektleiter, Autor, Leser).

- ***C-LAB-Text:*** Ein C-LAB-Text ist ein beliebiger hierarchisch in Abschnitte unterteilter Text. Der in der Hierarchie am höchsten stehende Abschnitt ist der Text selbst. Jeder Abschnitt ist einem oder mehreren Benutzern zugeordnet. Nur diese Benutzer sind berechtigt, den Abschnitt zu bearbeiten. Texte können beliebig viele Verweise auf Ressourcen enthalten. Ein Projekt enthält immer nur genau einen Text.

- ***C-LAB-Ressource:*** Eine C-LAB-Ressource ist ein beliebiges Dokument. Dies kann eine Grafik oder ein Video auf demselben Server sein. Ein Text kann auf diese Ressourcen verweisen, um sie inhaltlich einzubinden. Ressourcen können aber auch unabhängig vom eigentlichen Text existieren. So könnten Quelldokumente (z.B. eine Sammlung von PDF-Dokumenten), die während der Erstellung des Textes von Bedeutung sind, dort abgelegt werden. Ein Projekt kann mehrere Ressourcen enthalten, diese können hierarchisch in Ordnern verwaltet werden. Ressourcen können kopiert, verschoben, umbenannt, gelöscht und gesperrt werden. Jedem Benutzer können unterschiedlich abgestufte Zugriffsrechte für eine Ressource gewährt bzw. entzogen werden.

- ***C-LAB-Projekt:*** Ein C-LAB-Projekt enthält den eigentlichen Text, Ressourcen, Layouts und Kommentare. Ein Server kann mehrere Projekte auf einmal verwalten, diese können hierarchisch in Ordnern verwaltet werden. Projekte können kopiert, verschoben, umbenannt, gelöscht und gesperrt werden. Jedem Benutzer können unterschiedliche Zugriffsrechte für ein Projekt gewährt bzw. entzogen werden.

- ***C-LAB-Layout:*** Ein C-LAB-Layout ist eine Formatierungsbeschreibung des Textes. Während im Text selbst alle Elemente (z.B. Überschriften, Aufzählungen, Tabellen, Zitate etc.) durch explizite Auszeichnung gekennzeichnet sind, erlaubt ein C-LAB-Layout die grafische Umsetzung. Ein Projekt kann mehrere Layouts enthalten, z.B: ein Druck-Layout und ein eBook-Layout. Layouts können kopiert, umbenannt, gelöscht und gesperrt werden. Jedem Benutzer können unterschiedlich abgestufte Zugriffsrechte für ein Layout gewährt bzw. entzogen werden.

- ***C-LAB-Kommentar:*** Ein C-LAB-Kommentar bietet die Möglichkeit, den Text oder einzelne Abschnitte zu kommentieren, ohne den Text an sich zu verändern. Jeder Kommentar kann mit Antworten versehen werden, die wiederum beantwortet werden können, so dass sich aus einem Kommentar auch eine Diskussion bilden kann. Kommentare können nur von Benutzern erzeugt werden, welche die Berechtigung dafür besitzen.

- ***C-LAB-System:*** Ein C-LAB-System basiert auf einer Client-Server-Architektur. Ein C-LAB-Server verwaltet Projekte und Benutzer. Jeder C-LAB-Client muss

sich am Server identifizieren und authentifizieren, um auf die für ihn freigegebenen Projekte zugreifen zu können.

Text, Ressourcen und Layouts können versioniert werden, so dass innerhalb eines Projektes stets auf ältere Versionen einzelner Projektinhalte zurückgegriffen werden kann.

5.3.3 C-LAB-Benutzer

Ein C-LAB-Benutzer identifiziert sich durch seinen Namen, authentifiziert sich durch sein Kennwort und wird anschließend durch den C-LAB-Server autorisiert. Die Autorisierung erfolgt aufgrund der dem Benutzer zugewiesenen Benutzerrichtlinien. Benutzergruppen erleichtert das Zuweisen von Benutzerrichtlinien. Wird ein Benutzer einer Benutzergruppe zugewiesen, so erhält er automatisch die Benutzerrichtlinien der Benutzergruppe.

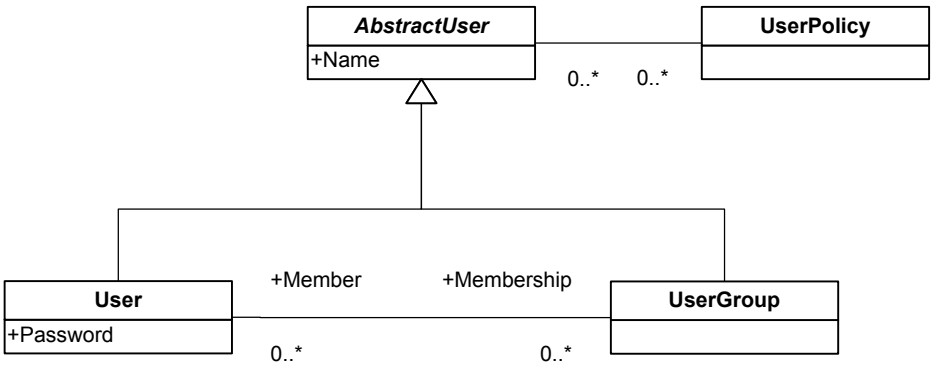

Abbildung 29: C-LAB-Benutzer und C-LAB-Benutzergruppen

C-LAB definiert lediglich vier Benutzerrichtlinien, da alle weiteren Richtlinien an konkrete Datenobjekte wie Projekte, Projektordner, Ressourcen oder Layouts gebunden sind.

Richtlinie	Beschreibung
Administrate	Erlaubt das Anlegen, Bearbeiten und Löschen von Benutzern und Benutzergruppen, sowie das Zuweisen von Richtlinien für einzelne Projekte und Projektordner.
BrowseProjects	Erlaubt das Auflisten aller Projekte und Projektordner, die nicht Teil eines anderen Projektordners sind.
CreateProjects	Erlaubt das Erzeugen eines neuen Projekts.
CreateProjectFolders	Erlaubt das Erzeugen eines neuen Projektordners.

Tabelle 9: C-LAB-Benutzerrichtlinien

5.3.4 C-LAB-Projekte

Ein C-LAB-Projekt besteht aus einem Text sowie einer optionalen Sammlung von Ressourcen, Layouts und Kommentaren. Projekte können hierarchisch in Projektordnern organisiert werden. Ein Projektordner kann beliebig viele Projekte sowie weitere Projektordner beinhalten. Für jedes Projekt und jeden Projektordner kann durch Zuweisen von Benutzern und Richtlinien die Zugriffsrechte festgelegt werden.

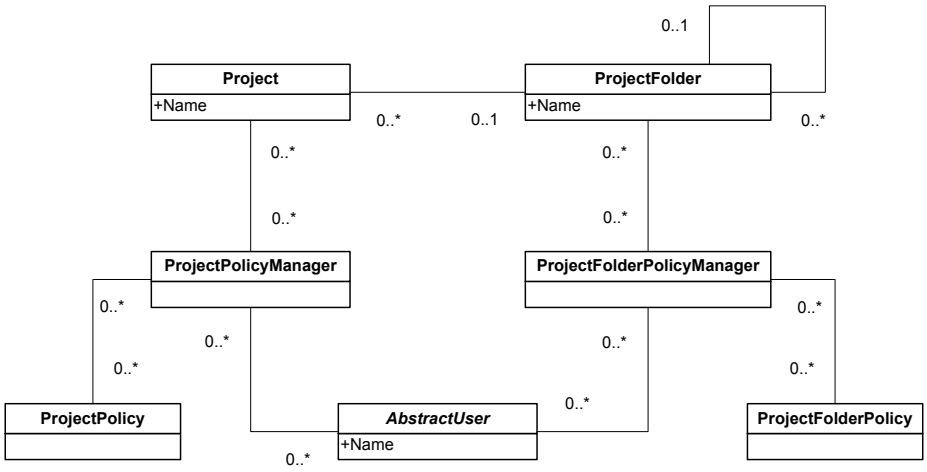

Abbildung 30: C-LAB-Projekte und C-LAB-Projektordner

C-LAB definiert für Projektordner die folgenden Richtlinien:

Richtlinie	Beschreibung
CreateProject	Erlaubt das Erzeugen eines neuen Projekts.
CreateProjectFolder	Erlaubt das Erzeugen eines neuen Projektordners.
DeleteProjectFolder	Erlaubt das Löschen eines bestehenden Projektordners.
LockProjectFolder	Erlaubt das exklusive Sperren eines bestehenden Projektordners für alle anderen Benutzer.
MoveProjectFolder	Erlaubt das Verschieben eines bestehenden Projektordners in einen anderen Projektordner.
OpenProjectFolder	Erlaubt das Öffnen eines Projektordners, um die enthaltenen Projekte und Projektordner auflisten zu können.
RenameProjectFolder	Erlaubt das Umbenennen eines bestehenden Projektordners.

Tabelle 10: Richtlinien für C-LAB-Projektordner

Für die in einem Projekt zusammengefasste Sammlung von Projektordnern, Ressourcen, Layouts und Kommentaren sind acht Richtlinien definiert:

Richtlinie	Beschreibung
AdministrateProject	Erlaubt Zuweisen von Richtlinien innerhalb des Projekts an andere Benutzer.
CopyProject	Erlaubt das Erstellen einer Kopie eines bestehenden Projekts.
CreateResource	Erlaubt das Erzeugen einer neuen Ressource.
DeleteProject	Erlaubt das Löschen eines bestehenden Projekts.
LockProject	Erlaubt das exklusive Sperren eines bestehenden Projekts für alle anderen Benutzer.
MoveProject	Erlaubt das Verschieben eines bestehenden Projekts in einen anderen Projektordner.
OpenProject	Erlaubt das Öffnen eines Projekts, um auf Text, Ressourcen, Layouts oder Kommentare zugreifen zu können.
RenameProject	Erlaubt das Umbenennen eines bestehenden Projekts.

Tabelle 11: Richtlinien für C-LAB-Projekte

5.3.5 C-LAB-Ressourcen

Eine C-LAB-Ressource ist ein Container für beliebige Dokumente. Dies können Grafiken, Videos oder Rohdaten für die Verwendung innerhalb des Textes sein oder aber zusätzliche Informationen, die für das gemeinsame Verständnis innerhalb des Projektes von Nutzen sein können. Ressourcen können, wie Projekte, hierarchisch in Ressource-Ordnern organisiert werden. Ein Ressource-Ordner kann beliebig viele Ressourcen sowie weitere Ressource-Ordner beinhalten.

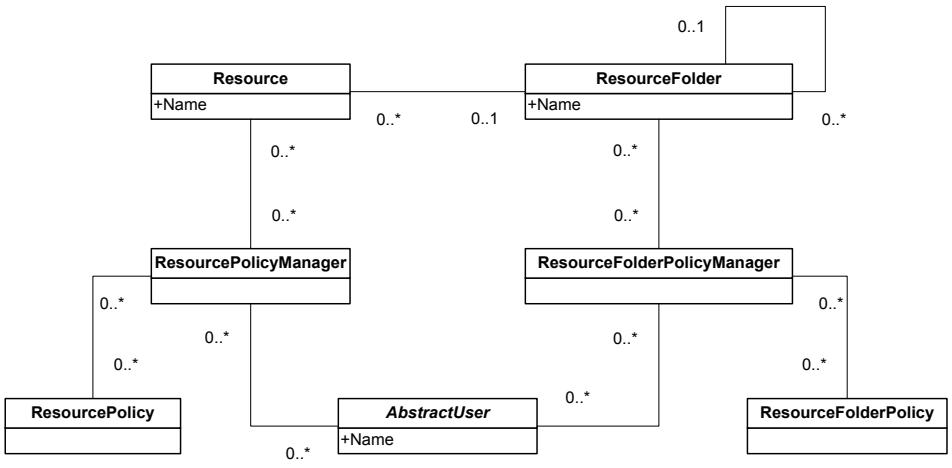

Abbildung 31: C-LAB-Ressourcen und C-LAB-Ressource-Ordner

Jeder Ressource bzw. jedem Ressource-Ordner können Richtlinien zugeweisen werden. C-LAB definiert die folgenden Richtlinien für Ressource-Ordner:

Richtlinie	Beschreibung
CreateResource	Erlaubt das Erzeugen einer neuen Ressource.
CreateResourceFolder	Erlaubt das Erzeugen eines neuen Ressource-Ordners.
DeleteResourceFolder	Erlaubt das Löschen eines bestehenden Ressource-Orders.
LockResourceFolder	Erlaubt das exklusive Sperren eines bestehenden Ressource-Ordners für alle anderen Benutzer.
MoveResourceFolder	Erlaubt das Verschieben eines bestehenden Ressource - Ordners in einen anderen Ressource-Ordner.
OpenResourceFolder	Erlaubt das Öffnen eines Ressource-Ordners, um die enthaltenen Ressourcen und Ressource-Ordner auflisten zu können.
RenameResourceFolder	Erlaubt das Umbenennen eines bestehenden Ressource-Ordners.

Tabelle 12: Richtlinien für C-LAB-Ressource-Ordner

C-LAB definiert die folgenden Richtlinien für Ressourcen:

Richtlinie	Beschreibung
CopyResource	Erlaubt das Erstellen einer Kopie einer bestehenden Ressource.
DeleteResource	Erlaubt das Löschen einer bestehenden Ressource.
LockResource	Erlaubt das exklusive Sperren einer bestehenden Ressource für alle anderen Benutzer.
MoveResource	Erlaubt das Verschieben einer bestehenden Ressource in einen anderen Ressource-Ordner.
OpenResource	Erlaubt das Öffnen einer bestehenden Ressource zum Anzeigen oder bearbeiten.
RenameResource	Erlaubt das Umbenennen einer bestehenden Ressource.

Tabelle 13: Richtlinien für C-LAB-Ressourcen

5.3.6 C-LAB-Layouts

Ein C-LAB-Layout enthält Formatierungsanweisungen für einen C-LAB-Text, um diesen in ein Publikationsformat zu überführen. Layouts können, wie Projekte, hierarchisch in Layout-Ordnern organisiert werden. Ein Layout-Ordner kann beliebig viele Layouts sowie weitere Layout-Ordner beinhalten.

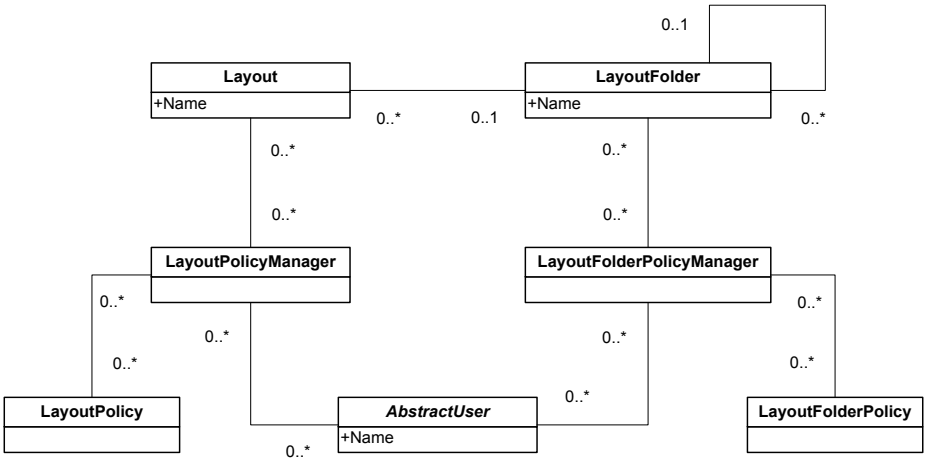

Abbildung 32: C-LAB-Layouts und C-LAB-Layout-Ordner

Jedem Layout bzw. jedem Layout-Ordner können Richtlinien zugeweisen werden. C-LAB definiert die folgenden Richtlinien für Layout-Ordner:

Richtlinie	Beschreibung
CreateLayout	Erlaubt das Erzeugen eines neuen Layouts.
CreateLayoutFolder	Erlaubt das Erzeugen eines neuen Layout-Ordners.
DeleteLayoutFolder	Erlaubt das Löschen eines existierenden Layout-Ordners.
LockLayoutFolder	Erlaubt das exklusive Sperren eines existierenden Layout-Ordners für alle anderen Benutzer.
MoveLayoutFolder	Erlaubt das Umplatzieren eines existierenden Layout-Ordners in einen anderen Ordner.
OpenLayoutFolder	Erlaubt das Öffnen eines existierenden Layout-Ordners, um die enthaltenen Layouts und Unterordner auflisten zu können.
RenameLayoutFolder	Erlaubt das Umbenennen eines existierenden Layout-Ordners.

Tabelle 14: Richtlinien für C-LAB-Layout-Ordner

C-LAB definiert die folgenden Richtlinien für Layouts:

Richtlinie	Beschreibung
CopyLayout	Erlaubt das Erstellen einer Kopie des Layouts.
DeleteLayout	Erlaubt das Löschen des Layouts.
LockLayout	Erlaubt das exklusive Sperren des Layouts für alle anderen Benutzer.
MoveLayout	Erlaubt das Umplazieren des Layouts in einen anderen

	Ordner.
OpenLayout	Erlaubt das Öffnen des Layouts zur Anwendung auf den C-LAB-Text des Projekts.
RenameLayout	Erlaubt das Umbenennen des Layouts.

Tabelle 15: Richtlinien für C-LAB-Layouts

5.3.7 C-LAB-Kommentare

Ein C-LAB-Kommentar ist immer mit einem bestimmten Textabschnitt in einem C-LAB-Text verbunden. Ein Kommentar kann einen Vorgänger-Kommentar und einen Nachfolge-Kommentar besitzen.

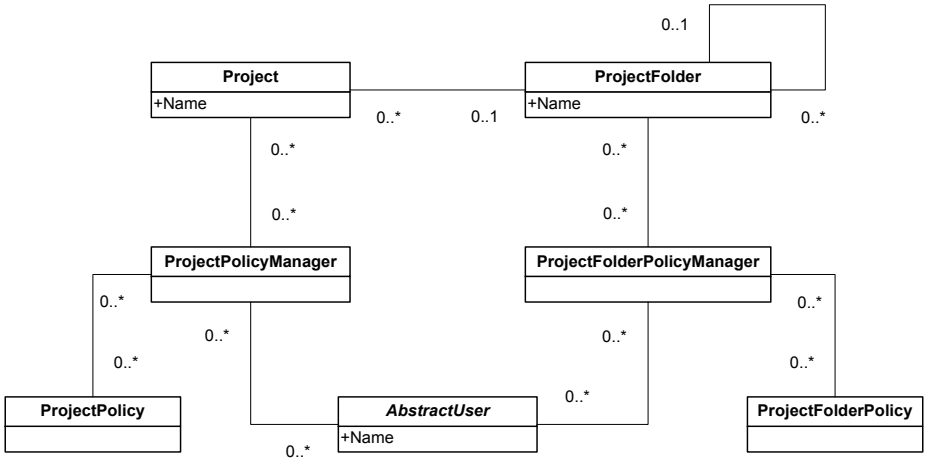

Abbildung 33: C-LAB-Kommentare

5.3.8 C-LAB-Text

Ein C-LAB-Text besteht aus einer Hierarchie von Abschnitten. Jeder Abschnitt enthält wiederum entweder weitere Abschnitte oder Text. Öffnet ein Benutzer einen Abschnitt zum Schreiben, so ist dieser Abschnitt und damit auch alle untergeordneten Abschnitte für andere Benutzer gesperrt. Diese können den Abschnitt zwar zum Lesen öffnen, sie können aber nicht zur selben Zeit schreiben.

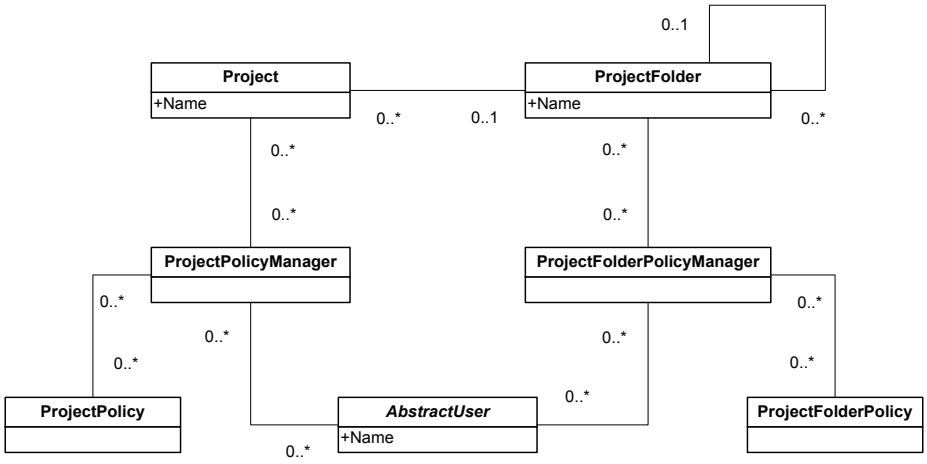

Abbildung 34: C-LAB-Text

5.3.9 C-LAB-Kommunikation

Alle C-LAB-Clients kommunizieren ausschließlich über einen C-LAB-Server miteinander. Dazu muss jeder C-LAB-Client wissen, wo sich der C-LAB-Server im Netzwerk befindet und wie er angesprochen wird.

Zur Kontaktaufnahme muss sich ein C-LAB-Client erst einmal identifizieren und authentifizieren. Dazu überträgt er seinen Benutzernamen und sein Kennwort. Der C-LAB-Server antwortet entweder mit einer Ablehnung (Benutzernamen oder Kennwort sind falsch) oder liefert eine Sitzungsnummer zurück. Mit Hilfe dieser Sitzungsnummer können zukünftige Anfragen des C-LAB-Client an den C-LAB-Server von Anfragen anderer C-LAB-Clients unterschieden werden, ohne dass der Benutzername und das Kennwort erneut übertragen werden muss.

Anschließend hat der C-LAB-Client folgende Möglichkeiten:

- Auflisten aller verfügbaren Projekte
- Öffnen eines Projektes
- Administrieren der Projekte (z.B. umbenennen oder löschen)

5.3.10 Diskussion

C-LAB setzt eine Client-Server-Architektur voraus, um Projekte und Benutzer effektiv verwalten zu können. Dies ist allerdings auch schon die einzige konkrete technische Vorgabe für ein C-LAB-kompatibles System. Die zentralen Verwaltung der C-LAB-Bausteine erfordert eine Realisierung als Client-Server-Lösung. Eine Dezentralisierung wäre dennoch möglich, in dem man mehrere C-LAB-Server zum Einsatz bringt, die sich untereinander mit einem geeigneten Replikationsmechanismus abgleichen. Allerdings macht eine derartige Skalierung nur bei sehr großen C-LAB-Projekten bzw. bei einer größeren Anzahl parallel laufender C-LAB-Projekte Sinn.

Darüber hinaus will C-LAB keine konkreten Implementationsvorgaben machen. Dies macht auch wenig Sinn, da sich bei der Entwicklung netzwerkbasierter Systeme in zunehmendem Maße diffuse Abgrenzungen herausbilden. Der in der Vergangenheit oft kolportierte Gegensatz Thin-Client vs. Thick-Client wird heute z.B. durch einen Technologiestreit zwischen konkurrierenden Web-Technologien wie HTML5, Silverlight und Flash überlagert. Hintergrund dieses Streits ist die Erkenntnis, dass klassisches HTML in der Regel keine praktikablen Möglichkeiten bietet, komplexe Softwaresysteme wie zum Beispiel eine Office-Anwendung sinnvoll zu implementieren. Die hier erwähnten Web-Technologien nehmen für sich in Anspruch, die Implementation komplexerer Web-basierter Clients zu ermöglichen, wenn auch eher mit Fokus auf der Erweiterung der grafisch/visuellen Funktionalität der Client-Anwendung. Eine Festlegung auf eine Technologie macht daher wenig Sinn und ist in Anbetracht des Ziels von C-LAB auch gar nicht nötig. C-LAB ist vielmehr ein Konzept, dass die dokumentenzentrierte Sichtweise auf Textverarbeitung durch eine benutzer- und projektorientierte Sichtweise ersetzt. Die sich daraus ergebenen Implikationen für ein konkretes System können die folgenden sein:

- Ein C-LAB-kompatibles Textverarbeitungssystem besteht aus mindestens zwei separaten Modulen: Einem Client und einem Server. Bei einer Web-Applikation mag diese Unterscheidung für den Benutzer nicht unmittelbar ersichtlich sein, sie ist aber logisch und praktisch gesehen da (Web-Server, Web-Browser).

- Die Kommunikation zwischen C-LAB-Server und C-LAB-Client sollte auf Basis eines offenen und standardisierten Kommunikationsprotokolls implementiert werden. Dies ist zwar nicht zwingend notwendig, aber angesichts der sich draus ergebenen Möglichkeiten (z.B. unterschiedliche Clients auf unterschiedlichen Betriebssystemen bzw. Endgeräten) dringend anzuraten. Ein Ansatz wäre es, die Protokollerweiterung TextDAV aus Kapitel 5.2 aufzugreifen und entsprechend weiterzuentwickeln.

- Das immer stärker in den Fokus rückende Thema des *Cloud-Computing* legt eine Implementierung unter Diensten wie *Microsoft Azur* nahe, da sie gleichzeitig eine elegante Lösung der Problematik des Hostings von C-LAB-Servern bietet. Ein C-LAB-Server muss schließlich im Netzwerk verfügbar sein, eine Vorgabe, die oft an fehlender Infrastruktur oder auch an fehlendem softwaretechnischem Know-how scheitert.

5.4 Schlussbetrachtung

Während TextDAV eine Erweiterung des bisherigen WebDAV-Standards darstellt, ist C-LAB gegenüber den konkreten Implementationen wie zum Beispiel Google Text und WinWord 2010 eine in dieser Arbeit neu vorgeschlagene Referenzarchitektur, die keine konkreten Implementierungsdetails vorgibt. C-LAB abstrahiert von einer konkreten Implementation analog zu TextDAV, das als Protokollspezifikation ebenfalls implementationsunabhängig ist.

Im Gegensatz zu WebDAV, Google Text und WinWord 2010 verwendet C-LAB analog zu TextDAV die Text- und Dokumentstruktur, um eine differenziertere Rechtevergabe und damit ein differenzierteres Zugriffssystem zu ermöglichen. Durch die Einbettung des Textverarbeitungssystems in die Gesamtarchitektur konnte das Datenmodell um Projekte, Layouts und Kommentare verfeinert werden. Die das kollaborative Arbeiten reflektierenden Rollen nehmen die differenzierte Dokumentensicht auf und setzen sie durch entsprechende Zugriffsrechte um.

Der C-LAB-Client könnte wahlweise als Desktop-Anwendungen als auch als Web-basierte Anwendung implementiert werden. Aktuelle Technologien wie *AJAX*, *Silverlight* oder *Adobe Air* erlauben mittlerweile Web-basierte Benutzeroberflächen zu gestalten, die denen herkömmlicher Desktop-Anwendungen in fast nichts nachstehen. Eine Implementierung als Desktop-Anwendungen hat allerdings den großen Vorteil, dass auch Offline-Szenarien unterstützt werden können. Der Idealfall wäre eine Implementierung beider Varianten. Die Desktop-Variante zum intensiven Arbeiten auf dem eigenen Arbeitsrechner und die Web-basierte Variante zum ortsunabhängigen Betrachten und Kommentieren. Auch WinWord 2010 verfügt über einen Windows-Client und einen Web-Client mit eingeschränktem Funktionsumfang.

Der Vorteil von C-LAB gegenüber TextDAV liegt kurz gesagt ins seiner Differenzierung bzw. seiner höheren Granularität. Bezogen auf wissenschaftliche Kollaboration könnte sich an dieser Stelle die Frage stellen, inwieweit eine noch weiter fortgeführte Verfeinerung Vorteile für ein KS-System bringen könnte. Denkbar wäre sicher, dass gerade bei sehr umfangreichen wissenschaftlichen Arbeiten, bei denen große Teams aus mehr als hundert Autoren Beiträge – möglicherweise auf unterschiedlichen Ebenen der Hierarchie – liefern, siehe als Beispiel ein internationales Physikprojekt mit Milliardenbudgets mit 1699 Autoren am CERN Center in Genf [Hofer et al. 08]), das Rollenmodell differenzierter modelliert werden sollte. Noch weiteren Differenzierungen scheint hier der schiere Umfang des Projekts bzw. die sehr hohe Anzahl von Autoren entgegenzustehen. Es treten hier verstärkt soziologische Probleme in den Vordergrund die so bei kleinen Teams eher unbedeutend sind: Je höher die Anzahl der Autoren desto eher versuchen einzelne Autoren sich zu profilieren, um aus der Masse hervorzustechen. Das widerspricht der charakteristischen Forderung innerhalb eines KS-Systems, sich mit anderen Autoren zu verständigen bzw. sich mit ihnen zu synchronisieren. Ein anderes berichtetes Problem liegt darin, dass die wissenschaftliche Glaubwürdigkeit in einem Dokument, das aus den Beiträgen vieler Co-Autoren besteht umso schwieriger beizubehalten ist je mehr Autoren daran mitarbeiten. Dies lässt sich wahrscheinlich weder durch ein differenzierteres Rollenkonzept noch durch umfangreichere Kommentierungen oder ein anderes des in dieser Arbeit vorgestellten methodischen Instrumentariums eines KS-Systems verhindern. Es zeigt sich bei großen Kollaborationsprojekten im wissenschaftlichen Bereich, dass die Lösung der soziologischen Probleme ab einer gewissen Teamgröße für die Kollaboration entscheidender ist als jede methodische oder modellbezogene Verbesserung in einem KS-System.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde das kollaborative Schreiben von Texten betrachtet, wobei die Fokussierung auf kollaboratives Arbeiten im technisch-wissenschaftlichen Umfeld lag, da diese Art von Zusammenarbeit dort häufiger als in anderen Bereichen vorkommt. Nach einer allgemeinen Einführung in diese Arbeit und Motivation für diese Arbeit, wurde in Kapitel 2 an Hand von Beispielen aus Wissenschaft und Wirtschaft die charakteristischen Anforderungen des kollaborativen Schreibens herausgearbeitet. Den Abschluss dieses Kapitels bildet eine allgemeingültige Taxonomie für kollaboratives Schreiben.

In Kapitel 3 wurden Textverarbeitungssysteme auf ihre kollaborativen Fähigkeiten hin untersucht. Es wurden zudem Anforderungen an ein idealtypisches kollaboratives Textverarbeitungssystem diskutiert, die als Vorgabe für die in Kapitel 5 weiter entwickelten KS-Systeme zu verstehen sind. Als Ergebnis konnte festgestellt werden, dass keines der untersuchten Systeme, auch nicht populäre Anwendungen wie Google Text und Microsoft Word, den Anforderungen an ein solches System in vollem Umfang entsprechen.

In Kapitel 4 wurden allgemeingültige Entwurfsmuster für KS-Systeme herausgearbeitet, die zum einen, korrespondierend zu Kapitel 2.4, eine Taxonomie für das Entwickeln und Beschreiben zukünftiger Software-Architekturen darstellen, zum anderen aber auch zeigen, dass es für viele Anforderungen an ein kollaboratives Textverarbeitungssystem bereits technische Lösungsansätze gibt. Es wurden passende Systemarchitekturen vorgestellt, das Management von Texten klassifiziert, das Konflikt-Management ausführlich diskutiert sowie Möglichkeiten der Nachvollziehbarkeit von Textänderungen betrachtet.

Kapitel 5 beschreibt auf der Grundlage der vorherigen Betrachtungen zwei konkrete idealtypische Systemarchitekturen für KS-Systeme. Die zwei auf Grundlage der in Kapitel 4 erläuterten Entwurfsmuster in Kapitel 5 vorgestellten Software-Architekturen erlauben es, die in Kapitel 3 festgestellten Unzulänglichkeiten bisheriger Systeme zu vermeiden. Zum einen war dies die in dieser Arbeit neu eingeführte Protokollerweiterung TextDAV, welche auf dem bestehenden WebDAV-Protokoll basiert, zum anderen die neu entwickelte Referenzarchitektur C-LAB.

Zwei Forschungsergebnisse haben sich von Anfang an sehr schnell herauskristallisiert:

1. Das Thema „Kollaboratives Schreiben“ ist nicht neu, trotzdem hat meines Erachtens die Forschungsdynamik in den letzten Jahren etwas an Fahrt verloren. Die immer wieder vorgeschlagenen Anforderungsprofile unterscheiden sich nur unwesentlich von den bereits existierenden Anforderungsprofilen älterer Arbeiten aus den 90er Jahren. Konkrete Entwicklungen von KS-Systemen sind in der Regel nicht über einen Prototypenstatus hinausge-

kommen oder erlauben lediglich das Erstellen einfachster Texte. Dabei gilt gerade das Zeitalter des Web 2.0 als Kollaborationszeitalter. Dienste wie Skype, Twitter, Facebook oder Wikis sind mittlerweile Konsens in der Gesellschaft. Die Beispiele in den Kapiteln 2.2.1 und 2.3.1 zeigen recht deutlich, dass signifikanter Bedarf an kollaborativen Textverarbeitungssystemen besteht.

2. Es gibt zur Zeit eine überschaubare Anzahl von kollaborativen Textverarbeitungssystemen, die alle nicht den Anforderungen an KS gerecht werden, wenn es darum geht, umfangreiche wissenschaftliche Texte kollaborativ zu erstellen. Es existieren auch keine ausgewiesenen KS-Protokollstandards, die eine kollaborative Interoperabilität unter Textverarbeitungssystemen ermöglichen.

Insbesondere der in Punkt 1 erwähnte Prototypenstatus der in der Literatur erwähnten Systeme scheint mit einem geringen Maß an wissenschaftlicher Publikation einher zu gehen, was dazu führte, dass das Recherchieren von Büchern und wissenschaftlichen Artikeln recht aufwendig war. Man konnte sogar den Eindruck gewinnen, dass die Forschung auf diesem Gebiet sich gänzlich in die Entwicklungslaboratorien der großen Software-Konzerne verlagert hat. Die beiden großen Gegenspieler auf diesem Gebiet sind gegenwärtig sicherlich Google und Microsoft. Während Google die Textverarbeitung, wie auch alle weiteren Office-Anwendungen, von vornherein als kollaboratives Werkzeug mit reduziertem Funktionsumfang ansieht, versucht Microsoft seine seit vielen Jahren erfolgreiche herkömmliche Textverarbeitung durch kollaborative Funktionen anzureichern. Nun fördert Konkurrenz ohne Zweifel Innovationen, allerdings scheinen mir beide Anwendungen trotz alledem einen falschen Weg einzuschlagen: Alle modernen Textverarbeitungssysteme basieren auf der WYSIWYG-Prinzip, d.h. der Autor bekommt auf dem Bildschirm ein möglichst exaktes Abbild dessen präsentiert, was er später in publizierter Form auf Papier oder in einem passsenden elektronischen Format präsentieren bzw. selber lesen kann. Dieses Prinzip, das mehr oder weniger auf einer Schreibmaschinen-Metapher beruht und sich darüber hinausgehend funktional wesentlich darauf konzentriert, die endgültige Textform zu repräsentieren, war und ist hervorragend für den Einzelautor geeignet. Dieselbe Metapher ist jedoch ungeeignet für das kollaborative Zusammenwirken mehrerer Autoren, bei der die in der Regel umfangreicheren Texte ein besonderes Augenmerk auf Struktur, Inhalte und Verfahrensweisen (Workflows) erfordern. Eine Schreibmaschine kann nun mal nicht von mehreren Autoren gleichzeitig genutzt werden und auch der Einsatz mehrerer Schreibmaschinen stellt aus leicht nachvollziehbaren Gründen keine Lösung da. Ich plädiere daher für eine Abkehr von der mehr oder weniger ausschließlichen Fokussierung auf das WYSIWYG-Prinzip, hin zu einer Entkopplung von Funktionalitäten innerhalb eines Textverarbeitungssystems. Beispiele für eine Entkopplung wären:

- Inhalt und Layout müssen voneinander nicht nur strikt voneinander getrennt werden, wie dies schon bei Systemen wie LaTeX realisiert wurde, sie müssen vielmehr vollständig entkoppelt werden. Dies entspricht dem Rollenverständnis aus Kapitel 2.4.5. Erst wird der Text von den Autoren geschrieben, dann wird er von einem Layouter passend zum gewünschten Publikations-

format gesetzt. Das Schreiben sowie das Layouten des Textes können im Extremfall von zwei völlig unterschiedlichen Anwendungen durchgeführt werden.

- Das Schreiben und Abgleichen von Inhalten darf nicht dazu führen, dass alle Autoren die gleiche Software benutzen müssen. Textverarbeitungssysteme müssen interoperabel zueinander sein und die Fähigkeit besitzen, untereinander Teiltexte auszutauschen und korrekt zu synchronisieren.

- Dieser Entkopplungsgedanke kann natürlich noch weiter geführt werden, so dass am Ende ein große Menge kleiner Spezialanwendungen steht, die jeweils nur einen ganz bestimmten Aspekt (z.B. das Kommentieren von Texten) von KS ausfüllen, gleichzeitig aber mit anderen Teilanwendungen kollaborieren können. Es entsteht ein Gesamtsystem, dass sich aus vielen kleinen autonomen Systemen zusammensetzt. Nicht nur Autoren kollaborieren jetzt miteinander sondern auch ihre individuellen Anwendungen. Multiagentensysteme aus dem Bereich *Artificial Intelligence* verfolgen dieselbe Idee, wenn auch biologisch motiviert: Vorbild ist der Ameisenstaat, indem viele Ameisen (Agenten) durch das Ausführen relativ einfacher Aufgaben im Zusammenspiel relativ komplexe Aufgaben lösen können. Die Abstraktion einer KS-Umgebung in organisatorische Einheiten innerhalb der vorgestellten C-LAB-Architektur in Kapitel 5.3 ist bereits ein erster Schritt in diese Richtung.

Die Vorteile liegen auf der Hand:

- Jeder Autor kann genau das Werkzeug einsetzen, welches für ihn am effizientesten ist. Ein Naturwissenschaftler legt beispielsweise Wert auf das korrekte Erstellen von Formeln, ein Geisteswissenschaftler legt vielleicht mehr Wert auf die Unterstützung von Fußnoten und Zitaten.

- Unterschiedliche Werkzeuge können für unterschiedliche Betriebssysteme und Endgeräte erstellt werden. Beispielsweise können Textkommentierungen durchaus von mobilen Endgeräten (z.B. Smartphone oder Tablet-Computer) aus getätigt werden, während für das Schreiben von umfangreichen Teiltexten wohl doch eher eine reale Tastatur wünschenswert ist.

All dies ist jedoch nur möglich, wenn diese Spezialanwendungen sich untereinander auch verstehen. Und dies bedeutet, dass zunächst ein oder mehrere Protokollstandards hierfür erarbeitet werden müssen. Mit der Vorstellung von TextDAV in Kapitel 5.2 wurde gezeigt, wie ein derartiges Protokoll in Bezug auf Textsynchronisation aussehen könnte.

Die Zukunft wird zeigen, wie sich kollaboratives Schreiben weiterentwickelt. Ich bin allerdings der festen Überzeugung, dass eine Abkehr von monolithischen Systemen wie Microsoft Word oder Google Text nötig ist, um wirkliche Innovation ermöglichen. Kollaboratives Schreiben ist zu facettenreich, als dass ein System allein alle Bedürfnisse befriedigen kann. Das World Wide Web als die Blaupause aller Kollaborationsumgebungen hat gezeigt, dass der Erfolg im Wesentlichen auf der Schaffung

von allgemein akzeptierten Protokollstandards basiert. Die Betrachtung der technologischen Rahmenbedingungen darf allerdings nicht außer Acht lassen, dass kollaboratives Schreiben auch ein sozialer Prozess ist (siehe Beispiele in Kapitel 2). Auch die beste Software ist immer nur ein Werkzeug in den Händen des Benutzers. Gegen Misstrauen oder fehlende Motivation hilft keine Technik, hier hilft nur aktive Kommunikation unter den Mitwirkenden. Es ist allerdings davon auszugehen, dass in den nächsten Jahren eine neue Generation von jungen Menschen heranwächst, die computer-basierte Kollaboration als selbstverständlich betrachten und die von daher eine wesentlich größere Bereitschaft zur Umsetzung von KS mitbringen. Die aktuellen politischen Entwicklungen in der arabischen Welt scheinen darauf hinzudeuten, dass die Kollaboration einer neuen internetgeprägten Generation durchaus sehr effektiv sein kann, wenn auch in diesem Fall außerhalb der hier besprochenen KS-Systeme.

Anhang A: Abkürzungsverzeichnis

C-LAB	Software Architecture for Collaborative Writing
CMS	Content Management System
CW	Collaborative Writing
DMS	Dokumenten-Management-System
DOC	Dateiendung für Word-Dokumente (bis einschließlich Version 2003)
DTP	Desktop-Publishing
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
KS	Kollaboratives Schreiben
ODMA	Open Document Management API
OT	Operational Transformations
PDF	Portable Document Format
RFC	Request for Comments
RTF	Rich Text Format
TEI	Text Encoding Initiative
TextDAV	Text Extensions for WebDAV
WebDAV	Web-based Distributed Authoring and Versioning
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

Anhang B: Abbildungsverzeichnis

Anhang C: Tabellenverzeichnis

Anhang D: Literatur

Artikel und Bücher

[Barrett 09]
Daniel J. Barrett (2009): „MediaWiki“. O'Reilly.

[Buretta 97]
Marie Buretta (1997): „Data Replication: Tools and Techniques for Managing Distributed Information“ John Wiley & Sons, Inc.

[Collins-Sussman et al. 04]
Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato (2004): „Version Control with Subversion“. O'Reilly.

[Dusseault 04]
Lisa Dusseault (2004): „WebDAV: Next-Generation Collaborative Web Authoring “. Prentice Hall.

[Ebel et al. 06]
Hans F. Ebel, Claus Bliefert, Walter Greulich (2006): „Schreiben und Publizieren in den Naturwissenschaften“. (5. Auflage). WILEY-VCH Verlag.

[Ebersbach et al. 08]
Anja Ebersbach, Markus Glaser, Richard Heigl, Alexander Warta (2008): „Wiki: Kooperation im Web“. (2. Auflage). Springer-Verlag.

[Ede et al. 90]
Lisa S. Ede, Andrea A. Lunsford (1990): „Singular Texts/Plural Authors”. Carbondale: Southern University Press.

[Eisenberg 01]
Daniel Eisenberg (1992): „WORD PROCESSING (HISTORY OF)“. Encyclopedia of Library and Information Science, vol. 49 (New York: Dekker), pp. 268-78

[Ellis et al. 89]
C. A. Ellis and S. J. Gibbs (1989): „Concurrency control in groupware systems”. In Proc. of the ACM Conference on Management of Data, pp. 399-407.

[Fowler 03]
Martin Fowler (2003): „Patterns für Enterprise Application-Architekturen“. Mitp-Verlag.

[Gamma et al. 94]
Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides (1994): „Design Patterns: Elements of Reusable Object-Oriented Software“. Addison-Wesley Professional

[Götzer et al. 04]
Götzer, Schneiderath, Maier, Bohmelt, Komke (2004): „Dokumenten-

Management: Informationen im Unternehmen effizient nutzen“. (3. Auflage). dpunkt.verlag GmbH.

[Günther et al. 08]
Karsten Günther, Matthias Kalle Dalheimer (2008): „LATEX kurz & gut“. (3. Aktualisierte Auflage). O’Reilly Verlag.

[Hansemann et al. 02]
Uwe Hansemann, Riku Mettälä, Apratim Purakayastha, Peter Thompson (2002): „SyncML: Synchronizing and Managing Your Mobile Data“ Prentice Hall.

[Harold et al. 04]
Elliotte Rusty Harold, W. Scott Means (2004): „XML in a Nutshell“. (3. Auflage). O’Reilly.

[Hofer et al. 08]
Erik C. Hofer, Shawn McKee, Jeremy P. Birnholtz and Paul Avery (2008): “High-Energy Physics: The Large Collider Collaborations” in [Olson et al. 08], pp. 143-151

[Hopcroft et al. 88]
John E. Hopcroft, Jeffrey D. Ullman (1988): „Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie ”. Addison-Wesley.

[Ignat et al. 03]
Claudia-Lavinia Ignat, Moira C. Norrie (2003): „Customizable Collaborative Editor Relying on treeOPT Algorithm”. Proceedings of the Eighth Conference on European Conference on Computer Supported Cooperative Work.

[Ignat et al. 07]
Claudia-Lavinia Ignat, Gérald Oster, Pascal Molli, Hala Skaf-Molli (2007): „A Collaborative Writing Mode for Avoiding Blind Modifications”. Ninth International Workshop on Collaborative Editing Systems.

[Jones 95]
Steve Jones (1995): „Identification and use of guidelines for the design of computer supported collaborative writing tools”. Computer Supported Cooperative Work (CSCW) 3: 379-404.

[Kecher 05]
Christoph Kecher (2005): „UML 2.0“. Galileo Press GmbH.

[Kirby et al. 95]
Andrew Kirby, Tom Rodden (1995): „Contact: Support for Distributed Cooperative Writing”. Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work.

[König 06]
Peter König (2006): „Teamwork im Netz”. c't 20/06, S. 105.

[Lamport 78]
Leslie Lamport (1978): „Time, Clocks and the Ordering of Events in a Distributed System”. Communications of the ACM, 21(7), 558-565.

[Lowry et al. 04]
Paul Benjamin Lowry, Aaron Curtis, Michelle René Lowry (2004): „Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research and Practice". Journal of business communication 41:66-99.

[Matthews et al. 00]
Janice R. Matthews, John M. Bowen (2000): „Successful Scientific Writing: A Step-by-Step Guide for the Biological and Medical Sciences". (2. Edition). Cambridge University Press.

[McAlpine et al. 94]
Keith McAlpine, Paul Golder (1994): „A New Architecture for a Collaborative Authoring System". Computer Supported Cooperative Work (CSCW) 2: 159-174.

[Noël et al. 04]
Sylvie Noël, Jean-Marc Robert (2004): „Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like". Computer Supported Cooperative Work, Vol. 13, No. 1, pp. 63-89.

[Noël et al. 03]
Sylvie Noël, Jean-Marc Robert (2003): „How the Web is used to support collaborative writing". Behaviour & Information Technology, Vol. 22, No. 4, pp. 245-262.

[Olson et al. 08]
Gary M. Olson, Ann Zimmerman, Nathan Bos, Herausgeber (2008): „Scientific Collaboration on the Internet". MIT Press

[Posner et al. 92]
Ilona R. Posner, Ronald M. Baecker (1992): „How People Write Together". Proceedings 25th Hawaii International Conference on System Sciences, Vol IV, 127-138.

[Preston et al. 05]
Jon A. Preston, Sushil K. Prasad (2005): „A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing". 7th Intl. Workshop on Collaborative Editing Systems.

[Rui Li, Du Li 2007]
Rui Li; Du Li (2007): „A New Operational Transformation Framework for Real-Time Group Editors". *IEEE Transactions on Parallel and Distributed Systems* (IEEE Press) **18(3)** (3): 307--319.

[Shen et al. 04]
Haifeng Shen, Chun Ti Cheong (2004): „CoStarOffice: Towards a Flexible Platformindependent Collaborative Office System". 6 th International Workshop on Collaborative Editing Systems.

[Shen et al. 06]
C. Sun, S. Xia, D. Sun, D. Chen. H.F. Shen, W. Cai (2006): „Transparent adaptation of single-user applications for multi-user real-time collaboration". ACM Transactions on Computer-Human Interaction, Vol. 13, No.4, pp.531-582.

[Shrum et al. 2007]
Wesley Shrum, Joel Genuth, Ivan Chompalov (2007): „Structures of scientific collaboration". MIT Press

[Simsion 01]
Graeme C. Simsion (2001): „Data Modeling Essentials: Analysis, Design and Innovation". The Coriolis Group.

[Stahl 96]
Peter Stahl (1996): „Tustep für Einsteiger: Eine Einführung in das Tübinger System von Textverarbeitungs-Programmen". Verlag Königshausen & Neumann.

[Sun et al. 06]
David Sun, Chengzheng Sun (2006): „Operation Context and Contextbased Operational Transformation". Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work.

[Tammaro et al. 97]
S. G. Tammaro, J. N. Mosier, N. C. Goodwin, G. Spitz (1997): „Collaborative Writing Is Hard to Support: A Field Study of Collaborative Writing". Computer Supported Cooperative Work: The Journal of Collaborative Computing 6: 19–51.

[Vogel 09]
Vogel, Arnold, Chughtai, Ihler, Kehrer, Mehlig, Zdun (2009): „Software-Architektur: Grundlagen - Konzepte - Praxis". (2. Auflage) Spektrum Akademischer Verlag Heidelberg.

[Xia et al. 04]
Steven Xia, David Sun, Chengzheng Sun, David Chen, Haifeng Shen (2004): „Leveraging Single-user Applications for Multi-user Collaboration: the CoWord Approach". Proceedings of the 2004 ACM conference on Computer supported cooperative work.

Internetreferenzen

[Borgolte et al. 08]
Michael Borgolte, Daniel Burckhardt, Jens Eremie, Juliane Schiel (2008): „Mediävistik trifft Technik: Ungewöhnliche Grenzerfahrungen zwischen den Disziplinen"
http://forschung.hu-berlin.de/publikationen/spektrum/borgolte_108.pdf
(Abgerufen: 16.03.2011)

[ECMA 08]
ECMA (2008): „ *Office Open XML File Formats, 2nd edition (December 2008)"*
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-376,%20Second%20Edition,%20Part%201%20-%20Fundamentals%20And%20Markup%20Language%20Reference.zip
(Abgerufen: 16.03.2010)

[Good 07]
Robin Good (2007): „Collaborative Writing Tools And Technology: A Mini-Guide"

http://www.kolabora.com/news/2007/03/01/collaborative_writing_tools_and_technology.htm
(Abgerufen: 16.03.2011)

[Hill 03]
Benjamin Mako Hill (2003): „Collaborative Literary Creation and Control"
http://mako.cc/projects/collablit/writing/BenjMakoHill-CollabLit_and_Control.pdf
(Abgerufen: 16.03.2011)

[IEEE 1471-2000]
IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems –Description
http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html
(Abgerufen: 25.12.2009)

[Meier 03]
Martin Meier (2003): „ShaDoW. A Collaborative Shared Document Writer"
www.dcg.ethz.ch/theses/ss03/ShaDoW_report.pdf
(Abgerufen: 16.03.2011)

[MLOC 08]
Make-Literature-Online Community (2008): „What is Collaborative Writing?"
http://www.makeliterature.com/blog/what-is-collaborative-writing
(Abgerufen: 16.03.2011)

[O'Reilly 05]
Tim O'Reilly (2005): „What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software"
http://www.oreilly.de/artikel/web20.html
(Abgerufen: 16.03.2011)

[ODF 09]
OASIS (2009): „*Open Document Format for Office Applications (OpenDocument) Version 1.2, Part 1: Introduction and OpenDocument Schema*"
http://docs.oasis-open.org/office/v1.2/part1/cd04/OpenDocument-v1.2-part1-cd04.html
(Abgerufen: 16.03.2011)

[PrePra 05]
Jon A. Preston, Sushil K. Prasad (2005): „A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing"
http://www3.ntu.edu.sg/home/ASHFShen/iwce07/papers/iwces2005_hierarchical_locking.pdf
(Abgerufen: 28.06.2009)

[RFC 1945]
RFC *1945* (1996): „ *Hypertext Transfer Protocol - HTTP/1.0*"
http://tools.ietf.org/html/rfc1945
(Abgerufen: 25.12.2009)

[RFC 2616]
RFC 2616 (1999): „ *Hypertext Transfer Protocol - HTTP/1.1*"
http://tools.ietf.org/html/rfc2616
(Abgerufen: 25.12.2009)

[RFC 4918]
RFC 4918 (2007): „HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)"
http://tools.ietf.org/html/rfc4918
(Abgerufen: 25.12.2009)

[Schiel et al. 08]
Michael Borgolte, Juliane Schiel, Bernd Schneidmüller und Annette Seitz (2008): „Die Mediävistik testet Wege zu einer transkulturellen Europawissenschaft"
https://www2.hu-berlin.de/sppedia/index.php5/Mittelalter_im_Labor:Inhalt
(Abgerufen: 28.06.2009)

[TEI 07]
TEI Consortium (2007): „Text Encoding Initiative: P5 Guidelines"
http://www.tei-c.org/release/doc/tei-p5-doc/en/Guidelines.pdf
(Abgerufen: 16.03.2011)

[UMUC 09]
University of Maryland University Collage (2009): „Tips for collaborative writing"
http://www.umuc.edu/departments/omde/orientation/collaborativeWriting.pdf
(Abgerufen: 28.06.2009)

[Weiser 91]
Mark Weiser (1991): „The Computer for the 21st Century"
http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html
(Abgerufen: 16.03.2011)

[ZDV 09]
Zentrum für Datenverarbeitung der Universität Tübingen (2009): „Textdatenverarbeitung mit TUSTEP"
http://www.zdv.uni-tuebingen.de/tustep/tdv.html
(Abgerufen: 28.06.2009)